ADVANCED SCIENTIFIC LIBRARY
ASL C INTERFACE
User's Guide
<Basic Functions Vol.6>

PROPRIETARY NOTICE

PREFACE

This manual describes general concepts, functions, and specifications for use of the Advanced Scientific Library (ASL) C interface.

The manuals corresponding to this product consist of seven volumes, which are divided into the chapters shown below. This manual describes the basic functions, volume 6.

Basic Functions Volume 1

Chapter	Title	Contents
1	Introduction	Explanation of the organization of this manual, how to view each item, and usage limitations.
2	Storage Mode Conversion	Explanation of algorithms, method of using, and usage example of function related to storage mode conversion of array data.
3	Basic Matrix Algebra	Explanation of algorithms, method of using, and usage example of function related to basic calculations involving matrices.
4	Eigenvalues and Eigenvectors	Explanation of algorithms, method of using, and usage example of function related to **the standard eigenvalue problem** for real matrices, complex matrices, real symmetric matrices, Hermitian matrices, real symmetric band matrices, real symmetric tridiagonal matrices, real symmetric random sparse matrices, Hermitian random sparse matrices and **the generalized eigenvalue problem** for real matrices, real symmetric matrices, Hermitian matrices, real symmetric band matrices.

Basic Functions Volume 2

Chapter	Title	Contents
1	Introduction	Explanation of the organization of this manual, how to view each item, and usage limitations.
2	Simultaneous Linear Equations (Direct Method)	Explanation of algorithms, method of using, and usage example of function related to simultaneous linear equations corresponding to real matrices, complex matrices, positive symmetric matrices, real symmetric matrices, Hermitian matrices, real band matrices, positive symmetric band matrices, real tridiagonal matrices, real upper triangular matrices, and real lower triangular matrices.

Basic Functions Volume 3

Chapter	Title	Contents
1	Introduction	Explanation of the organization of this manual, how to view each item, and usage limitations.
2	Fourier Transforms and their applications	Explanation of algorithms, method of using, and usage example of function related to one-, two- and three-dimensional complex Fourier transforms and real Fourier transforms, one-, two- and three-dimensional convolutions, correlations, and power spectrum analysis, wavelet transforms, and inverse Laplace transforms.

Basic Functions Volume 4

Chapter	Title	Contents
1	Introduction	Explanation of the organization of this manual, how to view each item, and usage limitations.
2	Differential Equations and Their Applications	Explanation of algorithms, method of using, and usage example of function related to **ordinary differential equations initial value problems** for high-order simultaneous ordinary differential equations, implicit simultaneous ordinary differential equations, matrix type ordinary differential equations, stiff problem high-order simultaneous ordinary differential equations, simultaneous ordinary differential equations, first-order simultaneous ordinary differential equations, and high-order ordinary differential equations, and **ordinary differential equations boundary value problems** for high-order simultaneous ordinary differential equations, first-order simultaneous ordinary differential equations, high-order ordinary differential equations, high-order linear ordinary differential equations, and second-order linear ordinary differential equations, and **integral equations** for Fredholm's integral equations of second kind and Volterra's integral equations of first kind, and **partial differential equations** for two- and three-dimensional inhomogeneous Helmholtz equation.
3	Numerical Differentials	Explanation of algorithms, method of using, and usage example of function related to numerical differentials of one-variable functions and multi-variable functions.
4	Numerical Integration	Explanation of algorithms, method of using, and usage example of function related to numerical integration over a finite interval, semi-infinite interval, fully infinite interval, two-dimensional finite interval, and multi-dimensional finite interval.
5	Interpolations and Approximations	Explanation of algorithms, method of using, and usage example of function related to interpolations, surface interpolations, least squares approximations, least squares surface approximations, and Chebyshev's approximations.
6	Spline Functions	Explanation of algorithms, method of using, and usage example of function related to interpolation, smoothing, numerical derivatives, and numerical integrals using cubic splines, bicubic splines and B–splines.

Basic Functions Volume 5

Chapter	Title	Contents
1	Introduction	Explanation of the organization of this manual, how to view each item, and usage limitations.
2	Special Functions	Explanation of algorithms, method of using, and usage example of function related to Bessel functions, modified Bessel functions, spherical Bessel functions, functions related to Bessel functions, Gamma functions, functions related to Gamma functions, elliptic functions, indefinite integrals of elementary functions, associated Legendre functions, orthogonal polynomials, and other special functions.
3	Sorting and Ranking	Explanation and usage examples of function related to sorting and ranking.
4	Roots of Equations	Explanation of algorithms, method of using, and usage example of function related to roots of algebraic equations, nonlinear equations, and simultaneous nonlinear equations.
5	Extremal Problems and Optimization	Explanation of algorithms, method of using, and usage example of function related to minimization of functions with no constraints, minimization of the sum of the squares of functions with no constraints, minimization of one-variable functions with constraints, minimization of multi-variable functions with constraints, and shortest path problem.

Basic Functions Volume 6

Chapter	Title	Contents
1	Introduction	Explanation of the organization of this manual, how to view each item, and usage limitations.
2	Random Number Tests	Explanation and usage examples of function related to uniform random number tests, and distribution random number tests.
3	Probability Distributions	Explanation and usage examples of function related to continuous distributions and discrete distributions.
4	Basic Statistics	Explanation and usage examples of function related to basic statistics, variance-covariance and correlation.
5	Tests and Estimates	Explanation and usage examples of function related to interval estimates and tests.
6	Analysis of Variance and Design of Experiments	Explanation and usage examples of function related to one-way layout, two-way layout, multiple-way layout, randomized block design, Greco-Latin square method, cumulative Method.
7	Nonparametric Tests	Explanation and usage examples of function related to tests using χ^2 distribution and tests using other distributions.
8	Multivariate Analysis	Explanation and usage examples of function related to principal component analysis, factor analysis, canonical correlation analysis, discriminant analysis, cluster analysis.
9	Time Series Analysis	Explanation and usage examples of function related to autocorrelation, cross correlation, autocovariance, cross covariance, smoothing and demand forecasting.
10	Regression analysis	Explanation and usage examples of function related to linear Regression and nonlinear Regression.

Shared Memory Parallel Functions

Chapter	Title	Contents
1	Introduction	Explanation of the organization of this manual, how to view each item, and usage limitations.
2	Basic Matrix Algebra	Explanation of algorithms, method of using, and usage example of function related to obtain the product of real matrices and complex matrices.
3	Simultaneous Linear Equations (Direct Method)	Explanation of algorithms, method of using, and usage example of function related to simultaneous linear equations corresponding to real matrices, complex matrices, real symmetric matrices, and Hermitian matrices.
4	Simultaneous Linear Equations (Iteration Method)	Explanation of algorithms, method of using, and usage example of function related to simultaneous linear equations corresponding to real positive definite symmetric sparse matrices, real symmetric sparse matrices and real asymmetric sparse matrices.
5	Eigenvalues and Eigenvectors	Explanation of algorithms, method of using, and usage example of function related to the eigenvalue problem for real symmetric matrices and Hermitian matrices.
6	Fourier Transforms and their applications	Explanation of algorithms, method of using, and usage example of function related to one-, two- and three-dimensional complex Fourier transforms and real Fourier transforms, two- and three-dimensional convolutions, correlations, and power spectrum analysis.
7	Sorting	Explanation and usage examples of function related to sorting and ranking.

Document Version 3.0.0-230301 for ASL, March 2023

Remarks

(1) This manual corresponds to ASL 1.1. All functions described in this manual are program products.

(2) Proper nouns such as product names are registered trademarks or trademarks of individual manufacturers.

(3) This library was developed by incorporating the latest numerical computational techniques. Therefore, to keep up with the latest techniques, if a newly added or improved function includes the function of an existing function may be removed.

Contents

Chapter 1

INTRODUCTION

1.1 OVERVIEW

1.1.1 Introduction to The Advanced Scientific Library ASL C interface

Table 1−1 lists correspondences among product categories, functions of ASL and supported hardware platforms. Interfaces of those functions that have the same name and that belong to the same version of ASL are common among hardware platforms.

Table 1−1 Classification of functions included in ASL

Classification of Functions	Volume
Basic functions	Vol. 1-6
Shared memory parallel functions	Vol. 7

1.1.2 Distinctive Characteristics of ASL C interface

ASL C interface has the following distinctive characteristics.

(1) Functions are optimized using compiler optimization to take advantage of corresponding system hardware features.

(2) Special-purpose functions for handling matrices are provided so that the optimum processing can be performed according to the type of matrix (symmetric matrix, Hermitian matrix, or the like). Generally, processing performance can be increased and the amount of required memory can be conserved by using the special-purpose functions.

(3) Functions are modularized according to processing procedures to improve reliability of each component function as well as the reliability and efficiency of the entire system.

(4) Error information is easy to access after a function has been used since error indicator numbers have been systematically determined.

1.2 KINDS OF LIBRARIES

Numeric storage units of ASL C interface is 4-byte.

Table 1−2 Kinds of libraries providing ASL C interface

Size of variable(byte)		Declaration of arguments	Kind	Kind of library
integer	real			
4	8	int double	32bit integer Double-precision function	32bit integer library (link option: -lasl_sequential)
4	4	int float	32bit integer Single-precision function	
8	8	long double	64bit integer Double-precision function	64bit integer library (link option: -lasl_sequential_i64)
8	4	long float	64bit integer Single-precision function	

(∗1) Functions that appear in this documentation do not always support all of the four kinds of functions listed above. For those functions that do not support some of those function kinds, relevant notes will appear in the corresponding subsections.

(∗2) For compiling the program with functions in the 64-bit integer library, the option "`-DASL_LIB_INT64`" must be specified (See the Note (2) in 1.5).

1.3 ORGANIZATION

This section describes the organization of Chapters 2 and later.

1.3.1 Introduction

The first section of each chapter is a general introduction describing functions corresponding to each functions.

1.3.2 Organization of Function Description

The second section of each chapter sequentially describes the following topics for each function.

(1) Function

(2) Usage

(3) Arguments and return value

(4) Restrictions

(5) Error indicator (Return Value)

(6) Notes

(7) Example

Each item is described according to the following principles.

1.3.3 Contents of Each Item

(1) **Function**
Function briefly describes the purpose of the ASL C interface function.

(2) **Usage**
Usage describes the function name and the order of its arguments. In general, arguments are arranged as follows. When an argument is an address-passing variable, & is appended in front of the argument name.

ierr = function-name (input-arguments, input/output-arguments, output-arguments, isw, work);

isw is an input argument for specifying the processing procedure. ierr is a return value. In some cases, input/output arguments precede input arguments. The following general principles also apply.

- Array are placed as far to the left as possible according to their importance.
- The dimension of an array immediately follows the array name. If multiple arrays have the same dimension, the dimension is assigned as an argument of only the first array name. It is not assigned as an argument of subsequent array names.

(3) **Arguments and return value**
Arguments and return value are explained in the order described above in paragraph (2). The explanation format is as follows.

Arguments and return value	Type	Size	Input/Output	Contents
(a)	(b)	(c)	(d)	(e)

(a) Arguments and return value

Arguments and return value are explained in the order they are designated in the Usage paragraph.

(b) Type

Type indicates the data type of the argument. Any of the following codes may appear as the type.

I : Integer type

D : Double precision real

R : Real

Z : Double precision complex

C : Complex

There are 64-bit integer and 32-bit integer for integer type arguments. In a 32-bit (64-bit) integer type function, all the integer type arguments are 32-bit (64-bit) integer. In other words, kinds of libraries determine the sizes of integer type arguments (Refer to 1.4). In the user program, a 32-bit/64-bit integer type argument must be declared by `int`/ `long`, respectively.

(c) Size

Size indicates the required size of the specified argument. If the size is greater than 1, the required area must be reserved in the program calling this function.

1 : Indicates that argument is a variable.

n : Indicates that the argument is a vector (one-dimensional array) having n elements. The argument n indicating the size of this vector is defined immediately after the specified vector. However, if the size of a vector or array defined earlier, it is omitted following subsequently defined vectors or arrays. The size may be specified by only a numeric value or in the form of a product or sum such as $3 \times n$ or $n + m$.

(d) Input/Output

Input/Output indicates whether the explanation of argument contents applies to input time or output time.

i. When only "Input" appears

When the control returns to the program using this function, information when the argument is input is preserved. The user must assign input-time information unless specifically instructed otherwise. When the argument is a variable, the variable value must be passed.

ii. When only "Output" appears

Results calculated within the function are output to the argument. No data is entered at input time. When the argument is a variable, the variable address must be passed.

iii. When both "Input" and "Output" appear

Argument contents change between the time control passes to the function and the time control returns from the function. The user must assign input-time information unless specifically instructed otherwise. When the argument is a variable, the variable address must be passed.

iv. When "Work" appears

Work indicates that the argument is an area used when performing calculations within the function. A work area having the specified size must be reserved in the program calling this function. The contents of the work area may have to be maintained so they can be passed along to the next calculation.

(e) Contents

Contents describes information held by the argument at input time or output time.

- A sample Argument description follows.

Example

The statement of the function (ASL_dbgmlc, ASL_rbgmlc) that obtains the LU decomposition and the condition number of a real matrix is as follows.

Double precision:

```
ierr = ASL_dbgmlc  (a, lna, n, ipvt, &cond, w1);
```

Single precision:

```
ierr = ASL_rbgmlc  (a, lna, n, ipvt, &cond, w1);
```

The explanation of the arguments and return value is as follows.

Table 1−3 Sample Arguments and Return Value

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	a	Note {D∗ / R∗}	lna×n	Input	Real matrix A(two-dimensional array)
				Output	The matrix A decomposed into the matrix LU where U is a unit upper triangular matrix and L is a lower triangular matrix.
2	lna	I	1	Input	Adjustable dimension size of array a
3	n	I	1	Input	Order n of matrix A
4	ipvt	I*	n	Output	Pivoting information ipvt[i−1]: Number of the row exchanged with row i in the i-th step.
5	cond	{D / R}	1	Output	Reciprocal of the condition number
6	w1	{D∗ / R∗}	n	Work	Work area
7	ierr	I	1	Output	Error indicator (Return Value)

To use this function, arrays a, ipvt and w1 must first be allocated in the calling program so they can be used as arguments. a is a {double-precision / single-precision}Note real array of size [lna × n] , ipvt is an integer array of size n and w1 is a {double-precision / single-precision}real array of size n.

When the 64-bit integer version is used, all integer-type arguments (lna, n, ipvt and ierr) must be declared by using `long`, not `int`.

Note The entries enclosed in brace { } mean that the array should be declared double precision type when using function ASL_dbgmlc and real type when using function ASL_rbgmlc. Braces are used in this manner throughout the remainder of the text unless specifically stated otherwise.

Data must be stored in a, lna and n before this function is called. The LU decomposition and condition number of the assigned matrix are calculated with in the function, and the results are stored in array a and variable cond. In addition, pivoting information is stored in ipvt for use by subsequent functions.

ierr is a return value used to notify the user of invalid input data or an error that may occur during processing. If processing terminates normally, ierr is set to zero.

Since w1 is a work area used only within the function, its contents at input and output time have no special meaning.

(4) **Restrictions**

Restrictions indicate limiting ranges for function arguments.

(5) **Error indicator (Return Value)**

Each function has been given an error indicator as a return value. This error indicator, which has uniformly been given the variable name ierr, is placed at the end of the arguments. If an error is detected within the function, a corresponding value is output to ierr. Error indicator values are divided into five levels.

Table 1−4 Classification of Return Values

Level	Return value	Meaning	Processing result
Normal	0	Processing is terminated normally.	Results are guaranteed.
Warning	1000∼2999	Processing is terminated under certain conditions.	Results are conditionally guaranteed.
Fatal	3000∼3499	Processing is aborted since an argument violated its restrictions.	Results are not guaranteed.
	3500∼3999	Obtained results did not satisfy a certain condition.	Obtained results are returned (the results are not guaranteed).
	4000 or more	A fatal error was detected during processing. Usually, processing is aborted.	Results are not guaranteed.

(6) **Notes**

Notes describes ambiguous items and points requiring special attention when using the function.

(7) **Example**

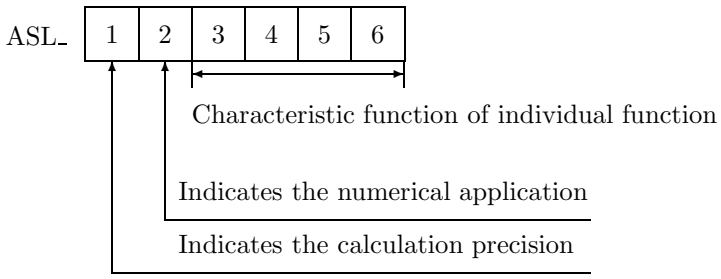
Here gives an example of how to use the function. Note that in some cases, multiple functions are combined in a single example. The output results are given in the 32-bit integer version, and may differ within the range of rounding error if the compiler or intrinsic functions are different.

In addition, when the 64-bit integer version library is used, the `long`-type conversion specification to be given to `printf` or `scanf` must be `%ld`. The source codes of examples in this document are included in User's Guide. Input data, if required, is also included in it. To build up an executable files by compiling these example source codes, they should be linked with this product library.

1.4 FUNCTION NAMES

The functions name of ASL C interface basic functions consists of ten characters with a prefix "ASL_" and ⟨six alphanumeric characters⟩.

Figure 1−1 Function Name Components



"1" in Figure 1−1 : The following eight letters are used to indicate the calculation precision.

d, w	Double precision real-type calculation
r, v	Single precision real-type calculation
z, j	Double precision complex-type calculation
c, i	Single precision complex-type calculation

However, the complex type calculations listed above do not necessarily require complex arguments.

"2" in Figure 1−1 : Currently, the following letters letterererererere are used to indicate the application field in the ASL C interface related products.

Letter	Application Field	Volume
a	Storage mode conversion	1
	Basic matrix algebra	1, 7
b	Simultaneous linear equations (direct method)	2, 7
c	Eigenvalues and eigenvectors	1, 7
f	Fourier transforms and their applications	3, 7
	Time series analysis	6
g	Spline function	4
h	Numeric integration	4
i	Special function	5
j	Random number tests	6
k	Ordinary differential equation (initial value problems)	4
l	Roots of equations	5
m	Extremum problems and optimization	5
n	Approximation and regression analysis	4, 6
o	Ordinary differential equations (boundary value problems), integral equations and partial differential equations	4
p	Interpolation	4
q	Numerical differentials	4

Letter	Application Field	Volume
s	Sorting and ranking	5, 7
x	Basic matrix algebra	1
	Simultaneous linear equations (iterative method)	7
1	Probability distributions	6
2	Basic statics	6
3	Tests and estimates	6
4	Analysis of variance and design of experiments	6
5	Nonparametric tests	6
6	Multivariate analysis	6

"3–6" in Figure 1−1 **:** These characters indicate the characteristic function of the individual function.

1.5 NOTES

(1) To use ASL C interface, the header file `asl.h`must be included.

(2) For compiling the program with functions in ASL C interface 64-bit integer library, the compile option "`-DASL_LIB_INT64`" must be specified. This option will activate the prototype declaration for 64-bit integer functions in the header file `asl.h`, and without the option "`-DASL_LIB_INT64`", those for 32-bit integer functions will be activated.

(3) The name "⟨6 lowercase letters⟩ following ASL_ " is reserved by ASL C interface.

(4) For using 64-bit integer library, you must use "`long`" for integer type declaration. Otherwise, use "`int`" for integer type declaration.

(5) Use the functions of double precision version whenever possible. They not only provide higher precision solutions but also are more stable than single precision versions, in particular, for eigenvalue and eigenvector problems.

(6) To suppress compiler operation exceptions, ASL C interface functions are set to so that they conform to the compiler parameter indications of a user's main program. Therefore, the main program must suppress any operation exceptions.

(7) The numerical calculation programs generally deal with operations on finite numbers of digits, so the precision of the results cannot exceed the number of operation digits being handled. For example, since the number of operation digits (in the mantissa part) for double-precision operations is on the order of 15 decimal digits, when using these floating point modes to calculate a value that mathematically becomes 1, an error on the order of 10^{-15} may be introduced at any time. Of course, if multiple length arithmetic is emulated such as when performing operations on an arbitrary number of digits, this kind of error can be controlled. However, in this case, when constants such as π or function approximation constants, which are fixed in double-precision operations, for example, are also to be subject to calculations that depend on the length of the multiple length arithmetic operations, the calculation efficiency will be worse than for normal operations.

(8) A solution cannot be obtained for a problem for which no solution exists mathematically. For example, a solution of simultaneous linear equations having a singular (or nearly singular) matrix for its coefficient matrix theoretically cannot be obtained with good precision mathematically. Numerical calculations cannot strictly distinguish between mathematically singular and nearly singular matrices. Of course, it is always possible to consider a matrix to be singular if the calculation value for the condition number is greater than or equal to an established criterion value.

(9) Generally, if data is assigned that causes a floating point exception during calculations (such as a floating point overflow), a normal calculation result cannot be expected. However, a floating point underflow that occurs when adding residuals in an iterative calculation is an exception to this.

(10) For problems that are handled using numerical calculations (specifically, problems that use iterative techniques as the calculation method), there are cases in which a solution cannot be obtained with good precision and cases in which no solution can be obtained at all, by a special-purpose function.

(11) Depending on the problem being dealt with, there may be cases when there are multiple solutions, and the execution result differs in appearance according to the compiler used or the computer or OS under which

the program is executed. For example, when an eigenvalue problem is solved, the eigenvectors that are obtained may differ in appearance in this way.

(12) The mark "DEPRECATED" denotes that the subroutine will be removed in the future. Use **ASL Unified Interface**, the higher performance alternative practice instead.

Chapter 2

RANDOM NUMBER TESTS

2.1 INTRODUCTION

This chapter describes the functions for testing random numbers.

The function for testing uniform random numbers can perform the following kinds of tests.

(1) Frequency one–dimensional test

(2) Frequency two–dimensional test

(3) Frequency three–dimensional test

(4) Run (ascending/descending) test

(5) Run (above/below) test

(6) Combination test

(7) Gap test

In addition, there are functions that perform frequency one–dimensional tests on each type of distribution random numbers.

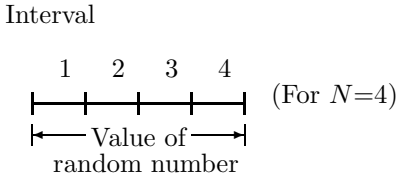
2.1.1 Explanation

(1) **Frequency one–dimensional test of uniform random numbers**

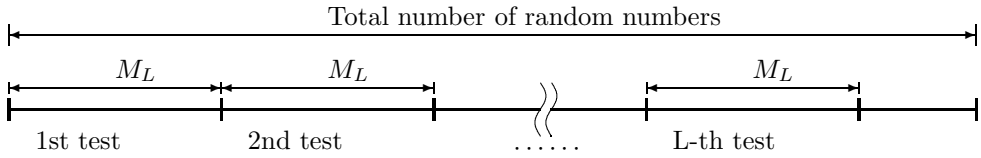
This is a frequency one–dimensional test of uniform random numbers in the range 0.0 through 1.0 that obtains the χ^2 value $X2$ shown below for each test, where N is the number of subdivisions and M_L is the number of random numbers used in a single test.

$$X2 = \sum_{i=1}^{N} \frac{(f_{P_i} - f_{T_i})^2}{f_{P_i}}$$

f_{T_i} : Total number of random numbers in interval i when the interval from 0.0 through 1.0 is subdivided into N equal parts.



The following figure shows which random numbers are used for each test.



f_{P_i} : $f_{P_i} = \dfrac{M_L}{N}$

The test is considered be passed if the following condition holds:
$X2 < \chi^2$ distribution percentile for the entered significance level

(2) **Frequency two–dimensional test of uniform random numbers**

This is a frequency two–dimensional test of uniform random numbers in the range 0.0 through 1.0 that obtains the χ^2 value $X2$ shown below for each test, where N is the number of subdivisions and M_L is the number of random numbers used in a single test.

$$X2 = \sum_{i=1}^{N}\sum_{j=1}^{N} \frac{(f_{P_{ij}} - f_{T_{ij}})^2}{f_{P_{ij}}}$$

$f_{T_{ij}}$: If the random numbers U_t, U_{t+1} are distributed as two–dimensional points in the X and Y coordinate system within a lattice created by subdividing the intervals from 0.0 through 1.0 on the X and Y axis into N equal parts, then $f_{T_{ij}}$ is the number of points within the corresponding lattice sector (See Fig. 2−1,2−2).

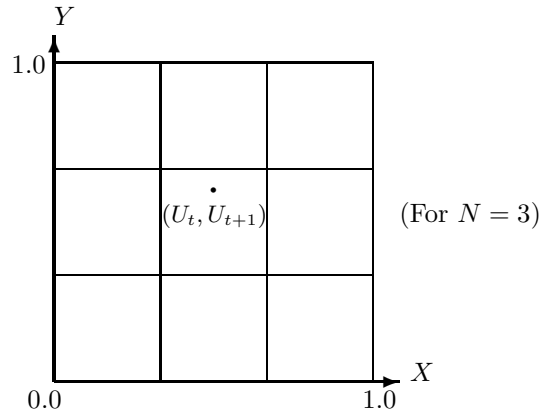


Figure 2−1

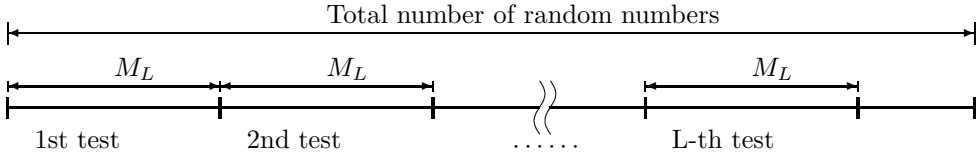


Figure 2−2

$$f_{P_{ij}} \quad : \quad f_{P_{ij}} = \frac{M_L}{N^2}$$

The test is considered to be passed if the following condition holds:
$X2 < \chi^2$ distribution percentile for the entered significance level

(3) **Frequency three–dimensional test of uniform random numbers**
This is a frequency three–dimensional test of uniform random numbers in the range 0.0 through 1.0 that obtains the χ^2 value $X2$ shown below for each test, where N is the number of subdivisions and M_L is the number of random numbers used in a single test.

$$X2 \quad = \quad \sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{k=1}^{N}\frac{(f_{P_{ijk}} - f_{T_{ijk}})^2}{f_{P_{ijk}}}$$

$f_{T_{ijk}}$: If the random numbers U_t, U_{t+1}, U_{t+2} are distributed as three–dimensional points in the X, Y and Z coordinate system within a lattice created by subdividing the intervals from 0.0 through 1.0 on the X, Y and Z axis into N equal parts, then $f_{T_{ijk}}$is the number of points within the corresponding lattice sector.(See Fig. 2−3 and 2−4)

$$f_{P_{ijk}} \quad : \quad f_{P_{ijk}} = \frac{M_L}{N^3}$$

The test is considered to be passed if the following condition holds:
$X2 < \chi^2$ distribution percentile for the entered significance level

(4) **Run (ascending/descending) test of uniform random numbers**
This is a run (ascending/descending) test of uniform random numbers in the range 0.0 through 1.0 that

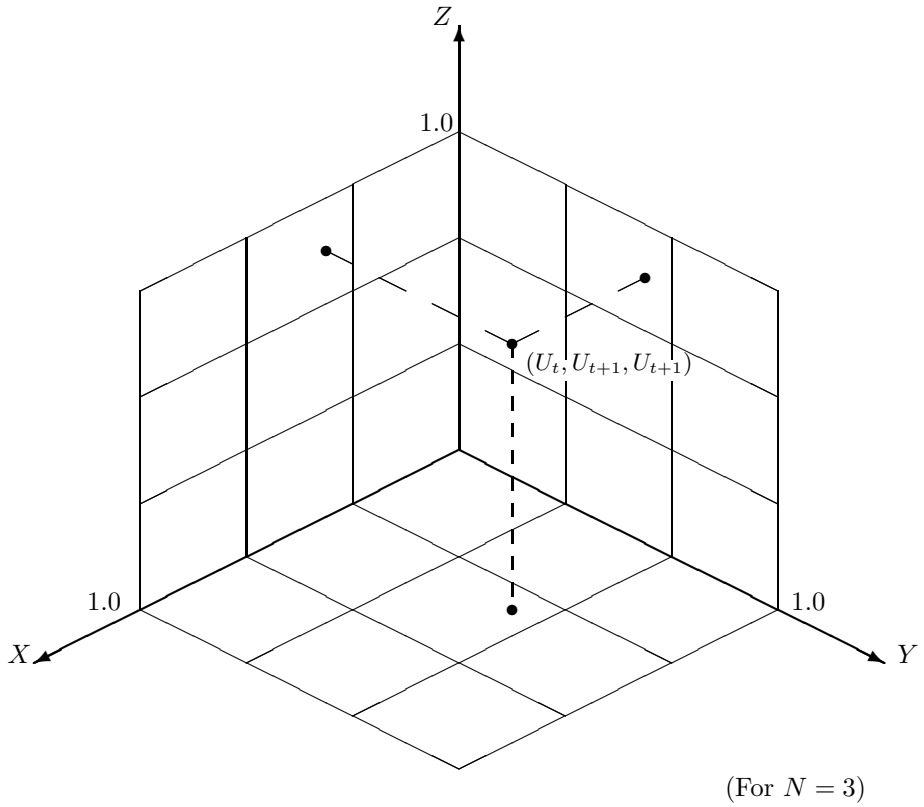


Figure 2−3

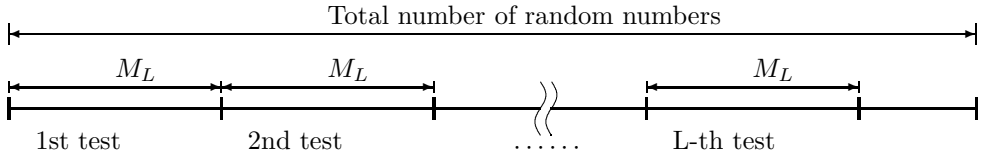


Figure 2−4

obtains the χ^2 value $X2$ shown below for each test, where N is the maximum run length and M_L is the number of random numbers used in a single test.

$$X2 = \sum_{i=1}^{N} \frac{(f_{P_i} - f_{T_i})^2}{f_{P_i}}$$

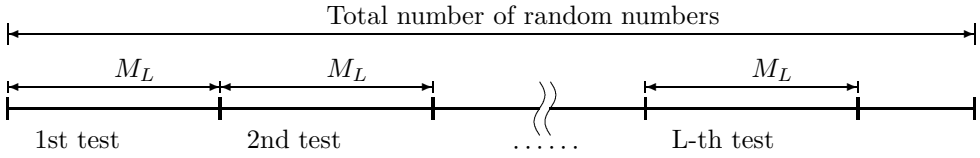
f_{T_i} : Number of ascending or descending runs of length i within the random number sequence. (However, for $i = N$, f_{T_i} is the number of ascending runs of length at least N.)

Example of an ascending run of length i

$$U_{t-1} > U_t < U_{t+1} < \cdots < U_{t+i} > U_{t+i+1}$$

The following figure shows which random numbers are used for each test.

f_{P_i} : $f_{P_i} = f'_{P_i}$ for $1 \leq i \leq N-1$



$$f_{P_N} = \sum_{k=N}^{M_L-1} f'_{P_k}$$

$$f'_{P_i} = 2 \times M_L \times \frac{i^2+3\times i+1}{(i+3)!} - 2 \times \frac{i^3+3\times i^2-i-4}{(i+3)!}$$

To improve test precision, this function uses a modified value for f_{P_i} defined by the following expression.

$$f^*_{P_i} = f_{P_i} \times \frac{\sum_{i=1}^{N} f_{T_i}}{\sum_{i=1}^{N} f_{P_i}}$$

The test is considered to be passed if the following condition holds:
$X2 < \chi^2$ distribution percentile for the entered significance level

(5) **Run (above/below) test of uniform random numbers**

This is a run (above/below) test of uniform random numbers in the range 0.0 through 1.0 that obtains the χ^2 value $X2$ shown below for each test, where N is the maximum run length and M_L is the number of random numbers used in a single test.

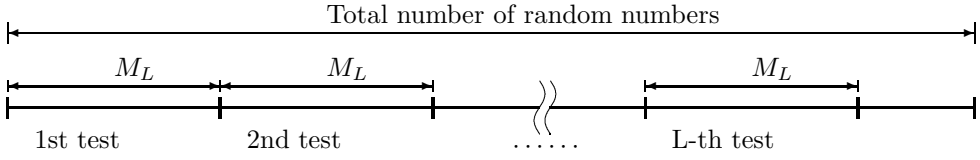
$$X2 = \sum_{i=1}^{N} \frac{(f_{P_i} - f_{T_i})^2}{f_{P_i}}$$

f_{T_i} : Number of runs above 0.5 of length i within the random number sequence.
(However, for $i = N$, f_{T_i} is the number of runs above 0.5 of length at least N.)

Example of a run above 0.5 of length 4

0.3, 0.6, 0.9, 0.7, 0.8, 0.1

The following figure shows which random numbers are used for each test.



f_{P_i} : $f_{P_i} = f'_{P_i}$ for $1 \le i \le N-1$

$$f_{P_N} = \sum_{k=N}^{M_L} f'_{P_k}$$

$$f'_{P_i} = \frac{M_L - i + 3}{2^{i+1}}$$

To improve test precision, this function uses a modified value for f_{P_i} defined by the following expression.

$$f^*_{P_i} = f_{P_i} \times \frac{\sum_{i=1}^{N} f_{T_i}}{\sum_{i=1}^{N} f_{P_i}}$$

The test is considered to be passed if the following condition holds:
$X2 < \chi^2$ distribution percentile for the entered significance level

(6) **Combination test of uniform random numbers**

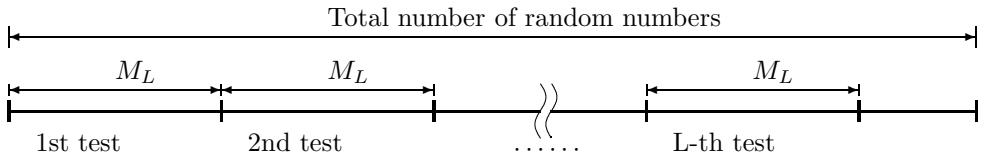
This is a combination test of uniform random numbers in the range 0.0 through 1.0. Conventionally, a combination test checks the number of "0" and "1" bits within the bit pattern of a single random number. However, in order to handle real random numbers, this function collects the register–length number of random numbers and tests how many of them are at least 0.5. The function obtains the χ^2 value $X2$ shown below for each test.

$$X2 = \sum_{i=0}^{N_B} \frac{(f_{P_i} - f_{T_i})^2}{f_{P_i}}$$

where N_B, which is the register length, is 32.

f_{T_i} : Number of groups of random numbers taken N_B at a time for which i numbers are at least 0.5.

The following figure shows which random numbers are used for each test.



$$f_{P_i} : f_{P_i} = \begin{pmatrix} N_B \\ i \end{pmatrix} \times \left(\frac{1}{2}\right)^{N_B} \times \sum_{i=1}^{N_B} f_{T_i} \times \left[\frac{M_L}{N_B}\right]$$

The actual subdivision is as follows.

$i = 0 \sim 8, 9, 10, \cdots, 22, 23, 24 \sim 32$

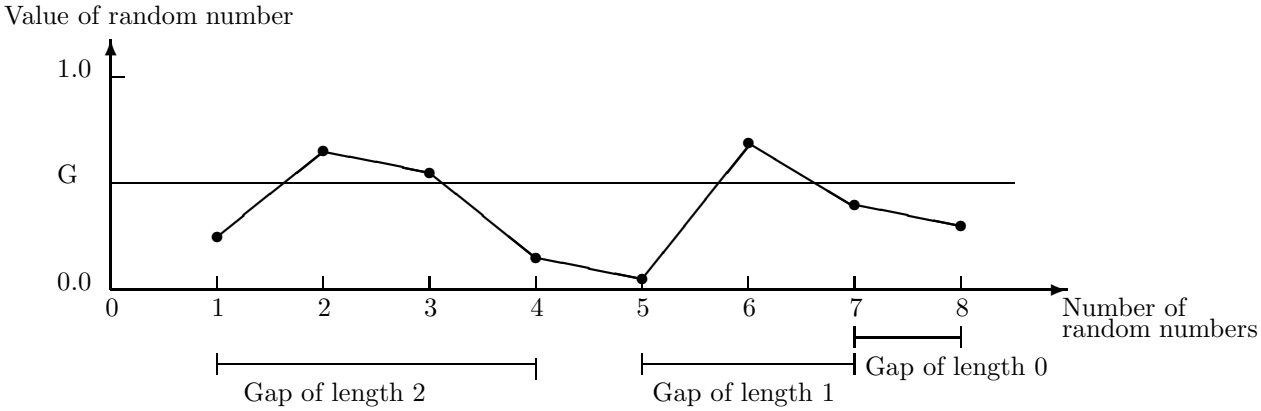
The test is considered to be passed if the following condition holds:
$X2 < \chi^2$ distribution percentile for the entered significance level

(7) **Gap test**

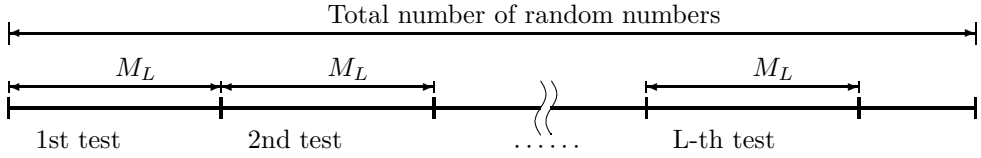
This is a gap test of uniform random numbers in the range 0.0 through 1.0 that obtains the χ^2 value $X2$ shown below for each test, where N is the maximum gap length, G is the gap value, and M_L is the number of random numbers used in a single test.

$$X2 = \sum_{i=0}^{N} \frac{(f_{P_i} - f_{T_i})^2}{f_{P_i}}$$

f_{T_i} : Number of gaps of length i in which sequences of random number that fall within the range 0.0 through G occur.



The following figure shows which random numbers are used for each test.



$$f_{P_i} \quad : \quad f_{P_i} = f'_{P_i} \quad \text{for} \quad 0 \leq i \leq N-1$$

$$f_{P_N} = \sum_{k=N}^{M_L-1} f'_{P_k}$$

$$f'_{P_i} = G \times (1.0 - G)^i \times M_L$$

To improve test precision, this function uses a modified value for f_{P_i} defined by the following expression.

$$f^*_{P_i} \quad = \quad f_{P_i} \times \frac{\sum_{i=0}^{N} f_{T_i}}{\sum_{i=0}^{N} f_{P_i}}$$

The test is considered to be passed if the following condition holds:
$X2 < \chi^2$ distribution percentile for the entered significance level

(8) **Tests of distribution random numbers**

A frequency one–dimensional test of distribution random numbers is performed.

(a) For a Continuous distribution

The test obtains the χ^2 value $X2$ shown below for each test, where U_L and U_P are the lower and upper limits of the test interval, N is the number of subdivisions, and M_L is the number of random numbers used in a single test.

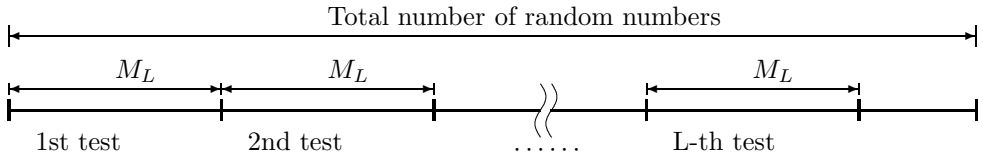
$$X2 \quad = \quad \sum_{i=1}^{N} \frac{(f_{P_i} - f_{T_i})^2}{f_{P_i}}$$

U_L and U_P are determined as follows depending on the type of distribution random numbers.

	Normal distribution Cauchy distribution Gumbel distribution Gamma distribution	Exponential distribution Weibull distribution
U_P	Entered test interval upper limit	Entered test interval upper limit
U_L	Entered test interval lower limit	0.0

f_{T_i} : Number of random numbers in interval i when the interval from U_L through U_P is subdivided into N equal parts. (Random numbers having values less than or equal to U_L are counted in interval 1 and those having values greater than or equal to U_P are counted in interval N.)

The following figure shows which random numbers are used in each test.



f_{P_i} : Expected frequency of random numbers in interval i.

If there is an expected frequency that a random number value will be less than or equal to U_L or greater than or equal to U_P, that expected frequency is added to the expected frequency of interval 1 or N, respectively.

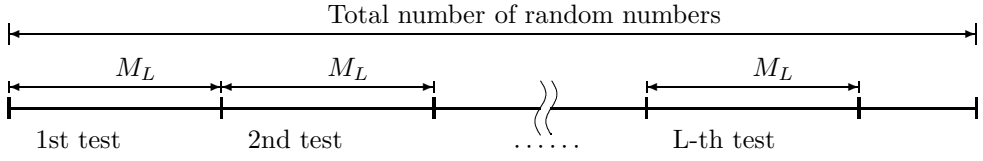
(b) For a Discrete distribution

The test obtains the χ^2 value $X2$ shown below for each test, where the integer U_P is the upper limit of the test interval and M_L is the number of random numbers used in a single test.

$$X2 = \sum_{i=0}^{U_P} \frac{(f_{P_i} - f_{T_i})^2}{f_{P_i}}$$

f_{T_i} : Number of random numbers having value i. (Random numbers having values greater than U_P are counted in numbercorresponding to U_P.)

The following figure shows which random numbers are used in each test.



f_{P_i} : Expected frequency of random numbers having value i. (If there is an expected frequency that a random number value will be greater than U_P, then that expected frequency is added to the expected frequency of U_P.)

2.1.2 Reference Bibliography

(1) Knuth, D. E. , "The Art of Computer Programming Volume II (2nd ed.) Seminumerical Algorithms/Random Numbers", Addison-Wesley (1981).

(2) Heringa, J. R. , Blote, H. W. J. , Compagner, A. , Int. J. Mod. Phys. C 3, 561 (1992)

(3) Kirpatrick, S. , Stoll, E. P. , "A Very Fast Shift–Register Sequence Random Number Generator", Journal of Computational Physics, Vol.40, pp.517–526, (1981).

2.2 UNIFORM RANDOM NUMBER TESTS

2.2.1 ASL_djteun, ASL_rjteun
Uniform Random Number Tests

(1) **Function**

Tests the given uniform random numbers in the range 0.0 through 1.0.

(2) **Usage**

Double precision:

```
ierr = ASL_djteun  (u, m, lt, n, g, alf, &k, x2, &cx, isw, wk);
```

Single precision:

```
ierr = ASL_rjteun  (u, m, lt, n, g, alf, &k, x2, &cx, isw, wk);
```

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	u	{D∗ / R∗}	m	Input	Random numbers
2	m	I	1	Input	Total number of random numbers
3	lt	I	1	Input	Test iteration count (See Note (a))
4	n	I	1	Input	isw=1, 2 or 3: Number of subdivisions isw=4 or 5: Maximum run length isw=6: Not used isw=7: Maximum gap length (See Note (b))
5	g	{D / R}	1	Input	Gap value (See Note (c))
6	alf	{D / R}	1	Input	Significance (%)
7	k	I*	1	Output	Number of passed tests
8	x2	{D∗ / R∗}	lt	Output	Test result χ^2 value
9	cx	{D∗ / R∗}	1	Output	χ^2 value for significance level

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
10	isw	I	1	Input	Processing switch isw=1: Frequency one-dimensional test isw=2: Frequency two-dimensional test isw=3: Frequency three-dimensional test isw=4: Run (ascending/descending) test isw=5: Run (above/below) test isw=6: Combination test isw=7: Gap test
11	wk	$\left\{\begin{matrix}\mathrm{D}* \\ \mathrm{R}*\end{matrix}\right\}$	See Contents	Work	Work area **Size**: isw=1: n isw=2: n^2 isw=3: n^3 isw=4 or 5: $2 \times n$ isw=6: 1 isw=7: $2 \times (n + 1)$
12	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) $1 \leq \mathrm{isw} \leq 7$

(b) $\mathrm{lt} \geq 1$

(c) $0.0 < \mathrm{alf} < 100.0$

(d) $\mathrm{m} \geq \mathrm{lt}$

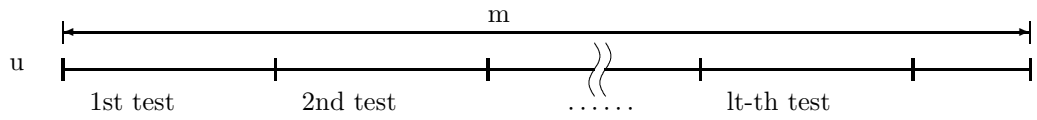
(e) For $\mathrm{isw} = 1 \sim 5$: $\mathrm{n} \geq 2$
For $\mathrm{isw} = 6$: $\mathrm{m} \geq 32 \times \mathrm{lt}$
For $\mathrm{isw} = 7$: $\mathrm{n} \geq 2$ and $0.0 < \mathrm{g} < 1.0$

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
1000	When $\mathrm{isw} \neq 6$, n was too large or m was too small. (See Note (b))	Processing continues. (Test precision gets worse.)
3000	Restriction (a), (b), (c), (d) or (e) was not satisfied.	Processing is aborted.
4000	$\mathrm{u[i-1]} < 0.0$ or $\mathrm{u[i-1]} > 1.0$, $\mathrm{i} = 1, \cdots, \mathrm{m}$.	

(6) **Notes**

(a) The following figure shows the relationship between the total number of random numbers m, the test iteration count lt, and the random numbers u used for each test.
$\lfloor \mathrm{m/lt} \rfloor$ random numbers u are taken at a time for each test, where $\lfloor x \rfloor$ represents the maximum integer that does not exceed x.



(b) For isw = 1, 2 or 3:
If the total number of random numbers M is too small or if the number of subdivisions n is too large, the expected frequency of random numbers in each interval decreases and test precision worsens.

Generally, the expected frequency F_{TN} should satisfy the following condition:

$$F_{TN} \geq 5.0$$

If this condition is not satisfied in these functions, then ierr=1000.

F_{TN} is calculated from the expressions shown below for each test, where $\lfloor x \rfloor$ represents the maximum integer that does not exceed x.

Frequency one-dimensional test (isw=1):

$$F_{TN} = \lfloor m/lt \rfloor \times \frac{1.0}{n}$$

Frequency two-dimensional test (isw=2):

$$F_{TN} = \lfloor m/lt \rfloor \times \frac{1.0}{2.0 \times n^2}$$

Frequency three-dimensional test (isw=3):

$$F_{TN} = \lfloor m/lt \rfloor \times \frac{1.0}{3.0 \times n^3}$$

For isw=4, 5 or 7:
If the total number of random numbers m is too small or if the maximum run length n or maximum gap length n is too large, the expected frequency for which the length is n decreases and test precision worsens.

Generally, the expected frequency F_{TN} should satisfy the following condition:

$$F_{TN} \geq 5.0$$

If this condition is not satisfied in these functions, then ierr = 1000. Expressions for calculating n, which is used as the test criterion, and its values are shown below for each test, where $\lfloor x \rfloor$ represents the maximum integer that does not exceed x.

For a run (ascending/descending) test (ISW=4)

$$n = \lfloor 1 + \log_{10} \lfloor m/lt \rfloor \rfloor$$

For example:

$\lfloor m/lt \rfloor$	100	1, 000	10, 000	100, 000	1, 000, 000
n	3	4	5	6	7

For a run (above/below) test (isw = 5)

$$n = \left\lfloor \frac{\log_{10}(\frac{\lfloor m/lt \rfloor}{10.0})}{\log_{10}(2)} \right\rfloor$$

For example:

⌊m/lt⌋	100	1, 000	10, 000	100, 000	1, 000, 000
n	3	6	9	13	16

For a gap test (isw = 7)

$$n = \left\lfloor \frac{\log_{10}\left(\frac{5.0}{G \times \lfloor m/lt \rfloor}\right)}{\log_{10}(1.0 - g)} \right\rfloor$$

For example, when g = 0.1:

⌊m/lt⌋	100	1, 000	10, 000	100, 000	1, 000, 000
n	6	28	50	72	93

(c) The gap value is used only when performing a gap test.
For explanation for gap value, see Section 2.1.1 Explanation (7).

(d) See the arguments table for the wk size.

(7) **Example**

(a) Problem
Perform a frequency one-dimensional test on 1000 uniform random numbers.

(b) Main program

```
/*      C interface example for ASL_djteun */

#include <stdio.h>
#include <stdlib.h>
#include <asl.h>

int main()
{
    double *u;
    int m=1000;
    int l=10;
    int n=10;
    double alf=1.0;
    int k;
    double *x2;
    double cx;
    double g=0.0;
    int isw=1;
    double *wk;
    int ierr;
    int i;

    int ix=1;
    int iy=1;

    printf( "    *** ASL_djteun ***\n" );
    printf( "\n     ** Input **\n\n" );

    u = ( double * )malloc((size_t)( sizeof(double) * m ));
    if( u == NULL )
    {
        printf( "no enough memory for array u\n" );
        return -1;
    }

    x2 = ( double * )malloc((size_t)( sizeof(double) * l ));
    if( x2 == NULL )
    {
        printf( "no enough memory for array x2\n" );
        return -1;
    }

    wk = ( double * )malloc((size_t)( sizeof(double) * n ));
    if( wk == NULL )
    {
        printf( "no enough memory for array wk\n" );
        return -1;
    }


    ierr = ASL_djufsp(m, &ix, &iy, u);

    printf( "\t  m = %6d\t\t  l = %8d\n", m, l );
    printf( "\t  n = %6d\t\talf = %8.3g\n", n, alf );
    printf( "\tisw = %6d\n", isw );

    ierr = ASL_djteun(u, m, l, n, g, alf, &k, x2, &cx, isw, wk);
```

```
    printf( "\n     ** Output **\n\n" );
    printf( "\tierr = %6d\n", ierr );

    printf( "\n\tNumber of Passed Test (k) = %6d\n", k );
    printf( "\n\t  Test No.       1       2       3       4       5" );
    printf(             "       6       7       8       9      10\n" );
    printf( "\tChi-Square\n" );
    printf( "\tValue (x2)" );
    for( i=0 ; i<l ; i++ )
    {
        printf( "%8.3g", x2[i] );
    }
    printf( "\n" );
    printf( "\n\tChi-Square Value for Percent Point (cx) = %8.3g\n", cx );

    free( u );
    free( x2 );
    free( wk );

    return 0;
}
```

(c) Output results

```
*** ASL_djteun ***

 ** Input **

  m =   1000          l =       10
  n =     10        alf =        1
isw =      1

 ** Output **

ierr =      0

Number of Passed Test (k) =     10

  Test No.       1       2       3       4       5       6       7       8       9      10
Chi-Square
Value (x2)     7.4    14.8     6.4     9.4     7.2       6    10.4     1.8    13.2    10.2

Chi-Square Value for Percent Point (cx) =     21.7
```

2.3 CONTINUOUS DISTRIBUTION RANDOM NUMBER TESTS

2.3.1 ASL_djteno, ASL_rjteno
Normal Distribution Random Number Test

(1) **Function**

Performs a frequency one-dimensional test on normal distribution random numbers.

(2) **Usage**

Double precision:

ierr = ASL_djteno (u, m, lt, n, alf, ul, up, am, sg, &k, x2, &cx, wk);

Single precision:

ierr = ASL_rjteno (u, m, lt, n, alf, ul, up, am, sg, &k, x2, &cx, wk);

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	u	{D∗ / R∗}	m	Input	Random numbers
2	m	I	1	Input	Total number of random numbers
3	lt	I	1	Input	Test iteration count (See Note (a))
4	n	I	1	Input	Number of subdivisions
5	alf	{D / R}	1	Input	Significance level (%)
6	ul	{D / R}	1	Input	Test interval lower limit (See Note (b))
7	up	{D / R}	1	Input	Test interval upper limit (See Note (b))
8	am	{D / R}	1	Input	Mean value
9	sg	{D / R}	1	Input	Standard deviation
10	k	I*	1	Output	Number of passed tests
11	x2	{D∗ / R∗}	lt	Output	Test result χ^2 value

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
12	cx	$\left\{\begin{matrix}\text{D}* \\ \text{R}*\end{matrix}\right\}$	1	Output	χ^2 value for significance level
13	wk	$\left\{\begin{matrix}\text{D}* \\ \text{R}*\end{matrix}\right\}$	$2 \times \text{n}$	Work	Work area
14	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) $\text{m} \geq \text{lt}$

(b) $\text{lt} \geq 1$

(c) $\text{n} \geq 2$

(d) $\text{up} > \text{ul}$

(e) $0.0 < \text{alf} < 100.0$

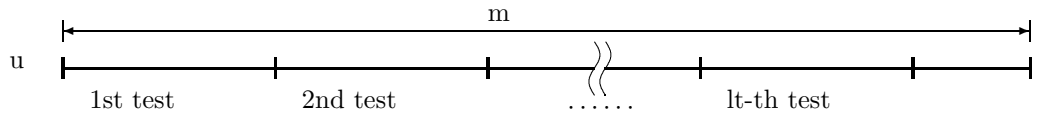
(f) $\text{sg} > 0.0$

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
1000	ul was too small, up was too large, or m was too small. (See Note (b))	Processing continues. (Test precision gets worse.)
3000	Restriction (a), (b), (c), (d), (e) or (f) was not satisfied.	Processing is aborted.

(6) **Notes**

(a) The following figure shows the relationship between the total number of random numbers m, the test iteration count lt, and the random numbers u used for each test.



$\lfloor \text{m/lt} \rfloor$ random numbers u are taken at a time for each test, where $\lfloor x \rfloor$ represents the maximum integer that does not exceed x.

(b) If the test range lower limit ul is too small or if the upper limit up is too large, then an extremely small expected frequency range is tested and test precision worsens.
If F_{Ti} is the expected frequency in each partial interval, then ierr = 1000 if the following condition occurs in the function:

$$\text{F}_{\text{Ti}} < 5 \qquad (\text{i} = 1, \cdots, \text{n})$$

Expressions for calculating ul and up, which are used as test criteria, are shown below.

$$\begin{aligned} \mathrm{ul} &= \mathrm{am} - \mathrm{sg} \times \mathrm{d} \\ \mathrm{up} &= \mathrm{am} + \mathrm{sg} \times \mathrm{d} \end{aligned}$$

where d is obtained from the following equation.

$$\mathrm{d} \times \mathrm{e}^{-\frac{\mathrm{d}^2}{2}} = \frac{5 \times \mathrm{n}}{\lfloor \mathrm{m/lt} \rfloor} \times \sqrt{\frac{\pi}{2}}$$

$\lfloor x \rfloor$ represents the maximum integer that does not exceed x.
Sample values of d are shown below.

$\frac{\mathrm{n}}{\lfloor \mathrm{m/lt} \rfloor}$	0.1	0.01	0.001	0.0001	0.00001
d	1.5	2.5	3.5	4.0	4.5

(7) **Example**

(a) Problem
Perform the test 10 times with 10 subdivisions on 10000 normal distribution random numbers having mean 0.0 and standard deviation 1.0.

(b) Main program

```
/*      C interface example for ASL_djteno */

#include <stdio.h>
#include <stdlib.h>
#include <asl.h>

int main()
{
    double *u;
    int m=10000;
    int l=10;
    int n=10;
    double alf=1.0;
    double ul= -2.5;
    double up=2.5;
    double am=0.0;
    double sg=1.0;
    int k;
    double *x2;
    double cx;
    double *wk;
    int ierr;
    int i;

    int ix=1;
    int iy=1;

    printf( "    *** ASL_djteno ***\n" );
    printf( "\n     ** Input **\n\n" );

    u = ( double * )malloc((size_t)( sizeof(double) * m ));
    if( u == NULL )
    {
        printf( "no enough memory for array u\n" );
        return -1;
    }

    x2 = ( double * )malloc((size_t)( sizeof(double) * l ));
    if( x2 == NULL )
    {
        printf( "no enough memory for array x2\n" );
        return -1;
    }

    wk = ( double * )malloc((size_t)( sizeof(double) * (2*n) ));
    if( wk == NULL )
    {
        printf( "no enough memory for array wk\n" );
        return -1;
    }


    ierr = ASL_djdbno(m, am, sg, &ix, &iy, u);

    printf( "\t m = %8d\t\t  l = %8d\n", m, l );
```

```
    printf( "\t n = %8d\t\talf = %8.3g\n", n, alf );
    printf( "\tul = %8.3g\t\t up = %8.3g\n", ul, up );
    printf( "\tam = %8.3g\t\t sg = %8.3g\n", am, sg );

    ierr = ASL_djteno(u, m, l, n, alf, ul, up, am, sg, &k, x2, &cx, wk);


    printf( "\n     ** Output **\n\n" );
    printf( "\tierr = %6d\n", ierr );

    printf( "\n\tNumber of Passed Test (k) = %6d\n", k );
    printf( "\n\t  Test No.       1       2       3       4       5" );
    printf(             "       6       7       8       9      10\n" );
    printf( "\tChi-Square\n" );
    printf( "\tValue (x2)" );
    for( i=0 ; i<l ; i++ )
    {
        printf( "%8.3g", x2[i] );
    }
    printf( "\n" );
    printf( "\n\tChi-Square Value for Percent Point (cx) = %8.3g\n", cx );

    free( u );
    free( x2 );
    free( wk );

    return 0;
}
```

(c) Output results

```
*** ASL_djteno ***

 ** Input **

 m =    10000         l =       10
 n =       10       alf =        1
ul =     -2.5        uu =      2.5
am =        0        sg =        1

 ** Output **

ierr =      0

Number of Passed Test (k) =     10

  Test No.       1       2       3       4       5       6       7       8       9      10
Chi-Square
Value (x2)     7.6    8.06    6.97    13.4    8.99    12.8     7.4    10.1    16.4    8.23

Chi-Square Value for Percent Point (cx) =     21.7
```

2.3.2 ASL_djteex, ASL_rjteex
Exponential Distribution Random Number Test

(1) **Function**

Performs a frequency one-dimensional test on exponential distribution random numbers.

(2) **Usage**

Double precision:

```
ierr = ASL_djteex  (u, m, lt, n, alf, up, am, &k, x2, &cx, wk);
```

Single precision:

```
ierr = ASL_rjteex  (u, m, lt, n, alf, up, am, &k, x2, &cx, wk);
```

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	u	{D∗ / R∗}	m	Input	Random numbers
2	m	I	1	Input	Total number of random numbers
3	lt	I	1	Input	Test iteration count (See Note (a).)
4	n	I	1	Input	Number of subdivisions
5	alf	{D / R}	1	Input	Significance level (%)
6	up	{D / R}	1	Input	Test interval upper limit (See Note (b))
7	am	{D / R}	1	Input	Mean value
8	k	I*	1	Output	Number of passed tests
9	x2	{D∗ / R∗}	lt	Output	Test result χ^2 value
10	cx	{D∗ / R∗}	1	Output	χ^2 value for significance level
11	wk	{D∗ / R∗}	$2 \times$ n	Work	Work area
12	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) $\mathrm{m} \geq \mathrm{lt}$

(b) $\mathrm{lt} \geq 1$

(c) $\mathrm{n} \geq 2$

(d) $0.0 < \mathrm{alf} < 100.0$

(e) $\mathrm{am} > 0.0$

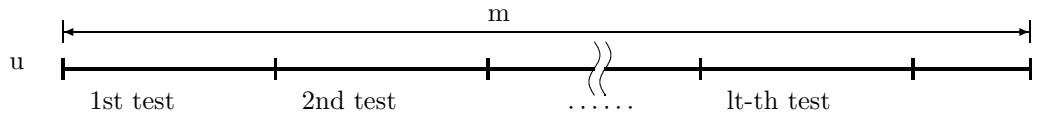
(f) $\mathrm{up} > 0.0$

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
1000	up was too large or m was too small. (See Note (b))	Processing continues. (Test precision gets worse.)
3000	Restriction (a), (b), (c), (d), (e) or (f) was not satisfied.	Processing is aborted.
4000	$\mathrm{u}[\mathrm{i}-1] < 0.0\,;\ \mathrm{i} = 1, \cdots, \mathrm{m}$	

(6) **Notes**

(a) The following figure shows the relationship between the total number of random numbers m, the test iteration count lt, and the random numbers u used for each test.



$\lfloor \mathrm{m/lt} \rfloor$ random numbers u are taken at a time for each test, where $\lfloor x \rfloor$ represents the maximum integer that does not exceed x.

(b) If the test range upper limit up is too large, then an extremely small expected frequency range is tested and test precision worsens.
If $\mathrm{F_{Ti}}$ is the expected frequency in each partial interval, then ierr = 1000 if the following condition occurs in this function:

$$\mathrm{F_{Ti}} < 5 \qquad (\mathrm{i} = 1, \cdots, \mathrm{n})$$

The value of up, which is used as a test criterion, is determined so that the following condition is satisfied.

$$\mathrm{up} \times \mathrm{e}^{-\frac{\mathrm{up}}{\mathrm{am}}} = \frac{5 \times \mathrm{n} \times \mathrm{am}}{\lfloor \mathrm{m/lt} \rfloor}$$

$\lfloor x \rfloor$ represents the maximum integer that does not exceed x.
Sample values of up are shown below for am=1.0.

$\frac{\mathrm{n}}{\lfloor \mathrm{m/lt} \rfloor}$	0.01	0.001	0.0001	0.00001
up	4	7	9	12

(7) **Example**

(a) Problem

Perform the test 10 times with subdivisions on 10000 exponential distribution random numbers having mean value 1.0.

(b) Main program

```
/*      C interface example for ASL_djteex */

#include <stdio.h>
#include <stdlib.h>
#include <asl.h>

int main()
{
    double *u;
    int m=10000;
    int l=10;
    int n=10;
    double alf=1.0;
    double up=4.0;
    double am=1.0;
    int k;
    double *x2;
    double cx;
    double *wk;
    int ierr;
    int i;

    int ix=1;
    int iy=1;

    printf( "    *** ASL_djteex ***\n" );
    printf( "\n     ** Input **\n\n" );

    u = ( double * )malloc((size_t)( sizeof(double) * m ));
    if( u == NULL )
    {
        printf( "no enough memory for array u\n" );
        return -1;
    }

    x2 = ( double * )malloc((size_t)( sizeof(double) * l ));
    if( x2 == NULL )
    {
        printf( "no enough memory for array x2\n" );
        return -1;
    }

    wk = ( double * )malloc((size_t)( sizeof(double) * (2*n) ));
    if( wk == NULL )
    {
        printf( "no enough memory for array wk\n" );
        return -1;
    }


    ierr = ASL_djdbex(m, am, &ix, &iy, u);

    printf( "\t m = %8d\t\t  l = %8d\n", m, l );
    printf( "\t n = %8d\t\talf = %8.3g\n", n, alf );
    printf( "\tup = %8.3g\t\t am = %8.3g\n", up, am );

    ierr = ASL_djteex(u, m, l, n, alf, up, am, &k, x2, &cx, wk);


    printf( "\n     ** Output **\n\n" );
    printf( "\tierr = %6d\n", ierr );

    printf( "\n\tNumber of Passed Test (k) = %6d\n", k );
    printf( "\t  Test No.       1       2       3       4       5" );
    printf(              "       6       7       8       9      10\n" );
    printf( "\tChi-Square\n" );
    printf( "\tValue (x2)" );
    for( i=0 ; i<l ; i++ )
    {
        printf( "%8.3g", x2[i] );
    }
    printf( "\n" );
    printf( "\n\tChi-Square Value for Percent Point (cx) = %8.3g\n", cx );

    free( u );
    free( x2 );
    free( wk );

    return 0;
}
```

(c) Output results

```
*** ASL_djteex ***

 ** Input **

 m =    10000          l =        10
 n =       10        alf =         1
up =        4         am =         1

 ** Output **

ierr =      0

Number of Passed Test (k) =      9
  Test No.       1       2       3       4       5       6       7       8       9      10
Chi-Square
Value (x2)      16    6.29    16.3    5.99    7.42    22.4     8.6    9.49     6.5    7.35

Chi-Square Value for Percent Point (cx) =     21.7
```

2.3.3 ASL_djtecc, ASL_rjtecc
Cauchy Distribution Random Number Test

(1) **Function**

Performs a frequency one-dimensional test on Cauchy distribution random numbers.

(2) **Usage**

Double precision:

```
ierr = ASL_djtecc  (u, m, lt, n, alf, ul, up, a, b, &k, x2, &cx, wk);
```

Single precision:

```
ierr = ASL_rjtecc  (u, m, lt, n, alf, ul, up, a, b, &k, x2, &cx, wk);
```

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	u	{D∗ / R∗}	m	Input	Random numbers
2	m	I	1	Input	Total number of random numbers
3	lt	I	1	Input	Test iteration count (See Note (a))
4	n	I	1	Input	Number of subdivisions
5	alf	{D / R}	1	Input	Significance level (%)
6	ul	{D / R}	1	Input	Test interval lower limit (See Note (b))
7	up	{D / R}	1	Input	Test interval upper limit (See Note (b))
8	a	{D / R}	1	Input	Value of the location parameter α.
9	b	{D / R}	1	Input	Value of the scale parameter β.
10	k	I*	1	Output	Number of passed tests
11	x2	{D∗ / R∗}	lt	Output	Test result χ^2 value
12	cx	{D∗ / R∗}	1	Output	χ^2 value for significance level

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
13	wk	$\begin{Bmatrix} \text{D}* \\ \text{R}* \end{Bmatrix}$	$2 \times \text{n} + 4$	Work	Work area
14	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) $\text{m} \geq \text{lt}$

(b) $\text{lt} \geq 1$

(c) $\text{n} \geq 2$

(d) $\text{up} > \text{ul}$

(e) $0.0 < \text{alf} < 100.0$

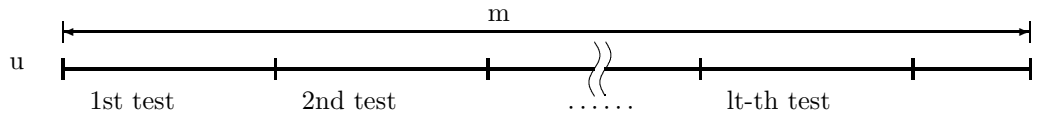
(f) $\text{bx} > 0.0$

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
1000	ul was too small, up was too large, or m was too small. (See Note (b))	Processing continues. (Test precision gets worse.)
3000	Restriction (a), (b), (c), (d), (e) or (f) was not satisfied.	Processing is aborted.

(6) **Notes**

(a) The following figure shows the relationship between the total number of random numbers m, the test iteration count lt, and the random numbers u used for each test.



$\lfloor \text{m/lt} \rfloor$ random numbers u are taken at a time for each test, where $\lfloor x \rfloor$ represents the maximum integer that does not exceed x.

(b) If the test range lower limit ul is too small or if the upper limit up is too large, then an extremely small expected frequency range is tested and test precision worsens.
If F_{Ti} is the expected frequency in each partial interval, then ierr = 1000 if the following condition occurs in the function:

$$\text{F}_{\text{Ti}} < 5 \qquad (\text{i} = 1, \cdots, \text{n})$$

Expressions for calculating ul and up, which are used as test criteria, are shown below.

$$\begin{aligned} \text{ul} &= \text{am} - \text{sg} \times \text{d} \\ \text{up} &= \text{am} + \text{sg} \times \text{d} \end{aligned}$$

where d is obtained from the following equation.

$$\mathrm{d} \times \mathrm{e}^{-\frac{\mathrm{d}^2}{2}} = \frac{5 \times \mathrm{n}}{\lfloor \mathrm{m/lt} \rfloor} \times \sqrt{\frac{\pi}{2}}$$

$\lfloor x \rfloor$ represents the maximum integer that does not exceed x.
Sample values of d are shown below.

$\frac{\mathrm{n}}{\lfloor \mathrm{m/lt} \rfloor}$	0.1	0.01	0.001	0.0001	0.00001
d	1.5	2.5	3.5	4.0	4.5

(7) **Example**

(a) Problem
Perform the test 10 times with 10 subdivisions on 10000 normal distribution random numbers with parameters $\alpha = 0.0$ and $\beta = 1.0$.

(b) Main program

```
/*      C interface example for ASL_djtecc */

#include <stdio.h>
#include <stdlib.h>
#include <asl.h>

int main()
{
    double *u;
    int m=10000;
    int l=10;
    int n=10;
    double alf=1.0;
    double ul=-2.5;
    double up=2.5;
    double a=0.0;
    double b=1.0;
    int k;
    double *x2;
    double cx;
    double *wk;
    int ierr;
    int i;

    int ix=1;
    int iy=1;

    printf( "    *** ASL_djtecc ***\n" );
    printf( "\n     ** Input **\n\n" );

    u = ( double * )malloc((size_t)( sizeof(double) * m ));
    if( u == NULL )
    {
        printf( "no enough memory for array u\n" );
        return -1;
    }

    x2 = ( double * )malloc((size_t)( sizeof(double) * l ));
    if( x2 == NULL )
    {
        printf( "no enough memory for array x2\n" );
        return -1;
    }

    wk = ( double * )malloc((size_t)( sizeof(double) * (2*n+4) ));
    if( wk == NULL )
    {
        printf( "no enough memory for array wk\n" );
        return -1;
    }

    ierr = ASL_djdbcc(m, a, b, &ix, &iy, u);

    printf( "\t m = %8d\t\t  l = %8d\n", m, l );
    printf( "\t n = %8d\t\talf = %8.3g\n", n, alf );
    printf( "\tul = %8.3g\t\t up = %8.3g\n", ul, up );
    printf( "\t a = %8.3g\t\t  b = %8.3g\n", a, b );

    ierr = ASL_djtecc(u, m, l, n, alf, ul, up, a, b, &k, x2, &cx, wk);

    printf( "\n     ** Output **\n\n" );
    printf( "\tierr = %6d\n", ierr );
```

```
    printf( "\n\tNumber of Passed Test (k) = %6d\n", k );
    printf( "\n\t  Test No.       1       2       3       4       5" );
    printf(               "       6       7       8       9      10\n" );
    printf( "\tChi-Square\n" );
    printf( "\tValue (x2)" );
    for( i=0 ; i<l ; i++ )
    {
        printf( "%8.3g", x2[i] );
    }
    printf( "\n" );
    printf( "\n\tChi-Square Value for Percent Point (cx) = %8.3g\n", cx );

    free( u );
    free( x2 );
    free( wk );

    return 0;
}
```

(c) Output results

```
*** ASL_djtecc ***

 ** Input **

 m =    10000         l =       10
 n =       10       alf =        1
ul =     -2.5        up =      2.5
 a =        0         b =        1

 ** Output **

ierr =      0

Number of Passed Test (k) =      9

  Test No.       1       2       3       4       5       6       7       8       9      10
Chi-Square
Value (x2)    11.9    7.69    14.1    13.1    4.72    25.3    12.2      12    4.01    11.3

Chi-Square Value for Percent Point (cx) =     21.7
```

2.3.4 ASL_djtegu, ASL_rjtegu
Gumbel Distribution Random Number Test

(1) **Function**

Performs a frequency one-dimensional test on Gumbel distribution random numbers.

(2) **Usage**

Double precision:

```
ierr = ASL_djtegu  (u, m, lt, n, alf, ul, up, a, b, &k, x2, &cx, wk);
```

Single precision:

```
ierr = ASL_rjtegu  (u, m, lt, n, alf, ul, up, a, b, &k, x2, &cx, wk);
```

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	u	{D∗ / R∗}	m	Input	Random numbers
2	m	I	1	Input	Total number of random numbers
3	lt	I	1	Input	Test iteration count (See Note (a))
4	n	I	1	Input	Number of subdivisions
5	alf	{D / R}	1	Input	Significance level (%)
6	ul	{D / R}	1	Input	Test interval lower limit (See Note (b))
7	up	{D / R}	1	Input	Test interval upper limit (See Note (b))
8	a	{D / R}	1	Input	Location parameter
9	b	{D / R}	1	Input	Scale parameter
10	k	I*	1	Output	Number of passed tests
11	x2	{D∗ / R∗}	lt	Output	Test result χ^2 value
12	cx	{D∗ / R∗}	1	Output	χ^2 value for significance level

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
13	wk	$\left\{\begin{matrix}\text{D}* \\ \text{R}*\end{matrix}\right\}$	$2 \times \text{n}$	Work	Work area
14	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) $\text{m} \geq \text{lt}$

(b) $\text{lt} \geq 1$

(c) $\text{n} \geq 2$

(d) $\text{up} > \text{ul}$

(e) $0.0 < \text{alf} < 100.0$

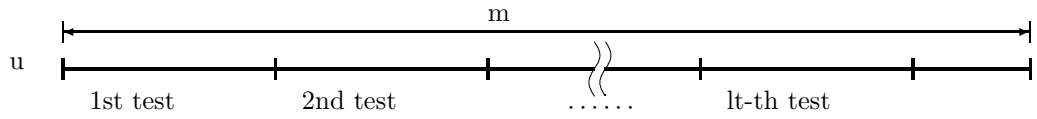
(f) $\text{b} > 0.0$

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
1000	ul was too small, up was too large, or m was too small. (See Note (b))	Processing continues. (Test precision gets worse.)
3000	Restriction (a) was not satisfied.	Processing is aborted.
3010	Restriction (b) was not satisfied.	
3020	Restriction (c) was not satisfied.	
3030	Restriction (d) was not satisfied.	
3040	Restriction (e) was not satisfied.	
3050	Restriction (f) was not satisfied.	

(6) **Notes**

(a) The following figure shows the relationship between the total number of random numbers m, the test iteration count lt, and the random numbers u used for each test.



$\lfloor \text{m/lt} \rfloor$ random numbers u are taken at a time for each test, where $\lfloor x \rfloor$ represents the maximum integer that does not exceed x.

(b) If the test range lower limit ul is too small or if the upper limit up is too large, then an extremely small expected frequency range is tested and test precision worsens.
If F_{Ti} is the expected frequency in each partial interval, then ierr = 1000 if the following condition occurs in the function:

$$\text{F}_{\text{Ti}} < 5 \qquad (\text{i} = 1, \cdots, \text{n})$$

Inequalities for calculating ul and up, which are used as test criteria, are shown below.

$$\min\left\{\exp\left(\frac{\mathrm{up}-\mathrm{a}}{\mathrm{b}}\right)\exp\left\lfloor-\exp\left(\frac{\mathrm{up}-\mathrm{a}}{\mathrm{b}}\right)\right\rfloor,\exp\left(\frac{\mathrm{ul}-\mathrm{a}}{\mathrm{b}}\right)\exp\left\lfloor-\exp\left(\frac{\mathrm{ul}-\mathrm{a}}{\mathrm{b}}\right)\right\rfloor\right\} \geq 5\times\frac{\mathrm{b}\times\mathrm{n}}{\lfloor\mathrm{m/lt}\rfloor\times(\mathrm{up}-\mathrm{ul})}$$

$\lfloor x\rfloor$ represents the maximum integer that does not exceed x.

(7) **Example**

(a) Problem

Perform the test 10 times with 10 subdivisions on 10000 Gumbel distribution random numbers having location parameter 0.0 and scale parameter 1.0.

(b) Main program

```
/*      C interface example for ASL_djtegu */

#include <stdio.h>
#include <stdlib.h>
#include <asl.h>

int main()
{
    double *u;
    int m=10000;
    int l=10;
    int n=10;
    double alf=5.0;
    double ul= -2.7;
    double up=2.7;
    double xa=1.0;
    double xb=1.0;
    int k;
    double *x2;
    double cx;
    double *wk;
    int ierr;
    int i;

    int ix=1;
    int iy=1;

    printf( "    *** ASL_djtegu ***\n" );
    printf( "\n     ** Input **\n\n" );

    u = ( double * )malloc((size_t)( sizeof(double) * m ));
    if( u == NULL )
    {
        printf( "no enough memory for array u\n" );
        return -1;
    }

    x2 = ( double * )malloc((size_t)( sizeof(double) * l ));
    if( x2 == NULL )
    {
        printf( "no enough memory for array x2\n" );
        return -1;
    }

    wk = ( double * )malloc((size_t)( sizeof(double) * (2*n+4) ));
    if( wk == NULL )
    {
        printf( "no enough memory for array wk\n" );
        return -1;
    }

    ierr = ASL_djdbgu(m, xa, xb, &ix, &iy, u);

    printf( "\t m = %8d\t\t  l = %8d\n", m, l );
    printf( "\t n = %8d\t\talf = %8.3g\n", n, alf );
    printf( "\tup = %8.3g\t\t ul = %8.3g\n", up, ul );
    printf( "\txa = %8.3g\t\t xb = %8.3g\n", xa, xb );

    ierr = ASL_djtegu(u, m, l, n, alf, ul, up, xa, xb, &k, x2, &cx, wk);

    printf( "\n     ** Output **\n\n" );
    printf( "\tierr = %6d\n", ierr );

    printf( "\n\tNumber of Passed Test (k) = %6d\n", k );
    printf( "\n\t  Test No.       1       2       3       4       5" );
    printf(                "       6       7       8       9      10\n" );
    printf( "\tChi-Square\n" );
```

```
        printf( "\tValue (x2)" );
        for( i=0 ; i<l ; i++ )
        {
            printf( "%8.3g", x2[i] );
        }
        printf( "\n" );
        printf( "\n\tChi-Square Value for Percent Point (cx) = %8.3g\n", cx );

        free( u );
        free( x2 );
        free( wk );

        return 0;
    }
```

(c) Output results

```
*** ASL_djtegu ***

 ** Input **

 m =    10000         l =       10
 n =       10       alf =        5
up =      2.7        ul =     -2.7
xa =        1        xb =        1

 ** Output **

ierr =      0

Number of Passed Test (k) =     10

  Test No.       1       2       3       4       5       6       7       8       9      10
Chi-Square
Value (x2)    7.09    8.18    10.4    7.99    9.67    14.7    12.1       4    4.67    10.9

Chi-Square Value for Percent Point (cx) =     16.9
```

2.3.5 ASL_djtewe, ASL_rjtewe
Weibull Distribution Random Number Tests

(1) **Function**

Performs a frequency one-dimensional test on Weibull distribution random numbers.

(2) **Usage**

Double precision:

```
ierr = ASL_djtewe  (u, m, lt, n, alf, up, a, b, &k, x2, &cx, wk);
```

Single precision:

```
ierr = ASL_rjtewe  (u, m, lt, n, alf, up, a, b, &k, x2, &cx, wk);
```

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	u	{D∗ / R∗}	m	Input	Random numbers
2	m	I	1	Input	Total number of random numbers
3	lt	I	1	Input	Test iteration count (See Note (a))
4	n	I	1	Input	Number of subdivisions
5	alf	{D / R}	1	Input	Significance level (%)
6	up	{D / R}	1	Input	Test interval upper limit (See Note (b))
7	a	{D / R}	1	Input	Shape parameter a
8	b	{D / R}	1	Input	Scale parameter b
9	k	I*	1	Output	Number of passed tests
10	x2	{D∗ / R∗}	lt	Output	Test result χ^2 value
11	cx	{D∗ / R∗}	1	Output	χ^2 value for significance level
12	wk	{D∗ / R∗}	$2 \times$ n	Work	Work area
13	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) $m \geq lt$

(b) $lt \geq 1$

(c) $n \geq 2$

(d) $0.0 < alf < 100.0$

(e) $up > 0.0$

(f) $a > 0.0$

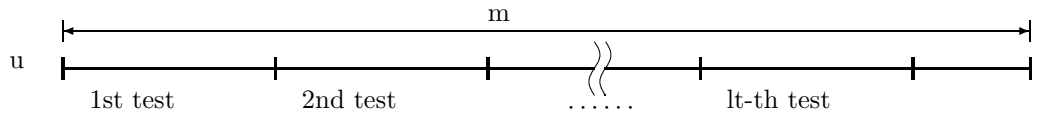
(g) $b > 0.0$

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
1000	up was too large or m was too small. (See Note (b))	Processing continues. (Test precision gets worse.)
3000	Restriction (a) was not satisfied.	Processing is aborted.
3010	Restriction (b) was not satisfied.	
3020	Restriction (c) was not satisfied.	
3030	Restriction (d) was not satisfied.	
3040	Restriction (e) was not satisfied.	
3050	Restriction (f) was not satisfied.	
3060	Restriction (g) was not satisfied.	

(6) **Notes**

(a) The following figure shows the relationship between the total number of random numbers m, the test iteration count lt, and the random numbers u used for each test.



$\lfloor m/lt \rfloor$ random numbers u are taken at a time for each test, where $\lfloor x \rfloor$ represents the maximum integer that does not exceed x.

(b) If the test range upper limit up is too large, then an extremely small expected frequency range is tested and test precision worsens.
If F_{Ti} is the expected frequency in each partial interval, then ierr = 1000 if the following condition occurs in this function:

$$F_{Ti} < 5 \qquad (i = 1, \cdots, n)$$

The value of up, which is used as a test criterion, is determined so that the following condition is satisfied.

$$up \times e^{-(\frac{up}{b})^a} = \frac{5 \times n}{\lfloor m/lt \rfloor}$$

$\lfloor x \rfloor$ represents the maximum integer that does not exceed x.
Sample values of up are shown below for a=2.0 and b=1.0 .

$\frac{n}{\lfloor m/lt \rfloor}$	0.01	0.001	0.0001	0.00001
up	1.9	2.5	2.9	3.3

(7) **Example**

(a) Problem
Perform the test 10 times with subdivisions on 10000 Weibull distribution random numbers having the shape parameter 2.0 and the scale parameter 1.0.

(b) Main program

```
/*      C interface example for ASL_djtewe */

#include <stdio.h>
#include <stdlib.h>
#include <asl.h>

int main()
{
    double *u, *x2, cx, *wk;
    int m=10000;
    int l=10;
    int n=10;
    double alf=5.0;
    double up=1.9;
    double a=2.0;
    double b=1.0;
    int k, ierr, i;

    int ix=1;
    int iy=1;

    printf( "    *** ASL_djtewe ***\n" );
    printf( "\n     ** Input **\n\n" );

    u = ( double * )malloc((size_t)( sizeof(double) * m ));
    if( u == NULL )
    {
        printf( "no enough memory for array u\n" );
        return -1;
    }

    x2 = ( double * )malloc((size_t)( sizeof(double) * l ));
    if( x2 == NULL )
    {
        printf( "no enough memory for array x2\n" );
        return -1;
    }

    wk = ( double * )malloc((size_t)( sizeof(double) * (2*n+4) ));
    if( wk == NULL )
    {
        printf( "no enough memory for array wk\n" );
        return -1;
    }

    ierr = ASL_djdbwe(m, a, b, &ix, &iy, u);

    printf( "\t m = %8d\t\t  l = %8d\n", m, l );
    printf( "\t n = %8d\t\talf = %8.3g\n", n, alf );
    printf( "\tup = %8.3g\n", up );
    printf( "\t a = %8.3g\t\t  b = %8.3g\n", a, b );

    ierr = ASL_djtewe(u, m, l, n, alf, up, a, b, &k, x2, &cx, wk);

    printf( "\n     ** Output **\n\n" );
    printf( "\tierr = %6d\n", ierr );

    printf( "\n\tNumber of Passed Test (k) = %6d\n", k );
    printf( "\t  Test No.       1       2       3       4       5" );
    printf(                 "       6       7       8       9      10\n" );
    printf( "\tChi-Square\n" );
    printf( "\tValue (x2)" );
    for( i=0 ; i<l ; i++ )
    {
        printf( "%8.3g", x2[i] );
    }
    printf( "\n" );
    printf( "\n\tChi-Square Value for Percent Point (cx) = %8.3g\n", cx );
```

```
    free( u );
    free( x2 );
    free( wk );

    return 0;
}
```

(c) Output results

```
*** ASL_djtewe ***

 ** Input **

 m =    10000         l =       10
 n =       10       alf =        5
up =      1.9
 a =        2         b =        1

 ** Output **

ierr =      0

Number of Passed Test (k) =      9
  Test No.       1       2       3       4       5       6       7       8       9      10
Chi-Square
Value (x2)    10.3    13.5    12.5     9.4    12.3    17.6    10.1    5.13    7.21    11.7

Chi-Square Value for Percent Point (cx) =     16.9
```

2.3.6 ASL_djtegm, ASL_rjtegm
Gamma Distribution Random Number Test

(1) **Function**

Performs a frequency one-dimensional test on gamma distribution random numbers.

(2) **Usage**

Double precision:

```
ierr = ASL_djtegm  (u, m, lt, ndiv, alf, ul, up, gamalp, &k, x2, &cx, wk);
```

Single precision:

```
ierr = ASL_rjtegm  (u, m, lt, ndiv, alf, ul, up, gamalp, &k, x2, &cx, wk);
```

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	u	{D∗ / R∗}	m	Input	Random numbers
2	m	I	1	Input	Total number of random numbers
3	lt	I	1	Input	Test iteration count (See Note (a)))
4	ndiv	I	1	Input	Number of subdivisions
5	alf	{D / R}	1	Input	Significance level (%)
6	ul	{D / R}	1	Input	Test interval lower limit (See Note (b))
7	up	{D / R}	1	Input	Test interval upper limit (See Note (b))
8	gamalp	{D / R}	1	Input	Shape parameter for gamma distribution
9	k	I*	1	Output	Number of passed tests
10	x2	{D∗ / R∗}	lt	Output	Test result χ^2 value
11	cx	{D∗ / R∗}	1	Output	χ^2 value for significance level
12	wk	{D∗ / R∗}	See Contents	Work	Work area **Size**: $2 \times \text{ndiv} + 4$
13	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) $m \geq lt$

(b) $lt \geq 1$

(c) $ndiv \geq 2$

(d) $up > ul$

(e) $0.0 < alf < 100.0$

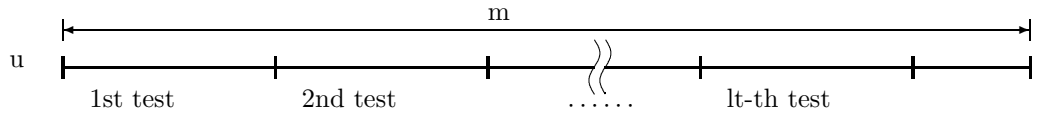
(f) $gamalp > 0.0$

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
1000	ul was too small, up was too large, or m was too small. (See Note (b))	Processing continues. (Test precision gets worse.)
3000	Restriction (a) was not satisfied.	Processing is aborted.
3010	Restriction (b) was not satisfied.	
3020	Restriction (c) was not satisfied.	
3030	Restriction (d) was not satisfied.	
3040	Restriction (e) was not satisfied.	
3050	Restriction (f) was not satisfied.	

(6) **Notes**

(a) The following figure shows the relationship between the total number of random numbers m, the test iteration count lt, and the random numbers u used for each test.



$\lfloor m/lt \rfloor$ random numbers u are taken at a time for each test, where $\lfloor x \rfloor$ represents the maximum integer that does not exceed x.

(b) If the test range lower limit ul is too small or if the upper limit up is too large, then an extremely small expected frequency range is tested and test precision worsens.
If F_{Ti} is the expected frequency in each partial interval, then ierr = 1000 if the following condition occurs in the function:

$$F_{Ti} < 5 \qquad (i = 1, \cdots, ndiv)$$

(7) **Example**

(a) Problem
Perform the test 10 times with 10 subdivisions on 10000 gamma distribution random numbers having the shape parameter $\alpha = 3.0$.

(b) Input data
$m = 10000$, $l = 10$, $ndiv = 10$, $alf = 5.0$, $ul = 0.0$, $up = 9.0$ and $gamalp = 3.0$.

(c) Main program

```
/*      C interface example for ASL_djtegm */

#include <stdio.h>
#include <stdlib.h>
#include <asl.h>

int main()
{
    double *u;
    int m=10000;
    int l=10;
    int n=10;
    double alf=5.0;
    double ul= 0.0;
    double up= 9.0;
    double alpha=3.0;
    int k;
    double *x2;
    double cx;
    double *wk;
    int ierr;
    int i;

    int ix=1;
    int iy=1;

    printf( "    *** ASL_djtegm ***\n" );
    printf( "\n     ** Input **\n\n" );

    u = ( double * )malloc((size_t)( sizeof(double) * m ));
    if( u == NULL )
    {
        printf( "no enough memory for array u\n" );
        return -1;
    }

    x2 = ( double * )malloc((size_t)( sizeof(double) * l ));
    if( x2 == NULL )
    {
        printf( "no enough memory for array x2\n" );
        return -1;
    }

    wk = ( double * )malloc((size_t)( sizeof(double) * (2*n+4) ));
    if( wk == NULL )
    {
        printf( "no enough memory for array wk\n" );
        return -1;
    }

    ierr = ASL_djdbgm(m, alpha, &ix, &iy, u);

    printf( "\t    m = %8d\t\t  l = %8d\n", m, l );
    printf( "\t    n = %8d\t\talf = %8.3g\n", n, alf );
    printf( "\t   ul = %8.3g\t\t up = %8.3g\n", ul, up );
    printf( "\talpha = %8.3g\n", alpha );

    ierr = ASL_djtegm(u, m, l, n, alf, ul, up, alpha, &k, x2, &cx, wk);

    printf( "\n     ** Output **\n\n" );
    printf( "\tierr = %6d\n", ierr );

    printf( "\n\tNumber of Passed Test (k) = %6d\n", k );
    printf( "\n\t  Test No.       1       2       3       4       5" );
    printf(              "       6       7       8       9      10\n" );
    printf( "\tChi-Square\n" );
    printf( "\tValue (x2)" );
    for( i=0 ; i<l ; i++ )
    {
        printf( "%8.3g", x2[i] );
    }
    printf( "\n" );
    printf( "\n\tChi-Square Value for Percent Point (cx) = %8.3g\n", cx );

    free( u );
    free( x2 );
    free( wk );

    return 0;
}
```

(d) Output results

```
*** ASL_djtegm ***

 ** Input **

    m =    10000          l =       10
    n =       10        alf =        5
```

```
   ul =        0         up =        9
alpha =        3

 ** Output **

ierr =      0

Number of Passed Test (k) =      9

  Test No.       1       2       3       4       5       6       7       8       9      10
Chi-Square
Value (x2)    3.19    7.94    9.18    9.71    1.86    6.58    20.3    8.62    8.74    6.38

Chi-Square Value for Percent Point (cx) =     16.9
```

2.3.7 ASL_djtelg, ASL_rjtelg
Logistic Distribution Random Number Test

(1) **Function**

Performs a frequency one-dimensional test on logistic distribution random numbers.

(2) **Usage**

Double precision:

```
ierr = ASL_djtelg  (u, m, lt, ndiv, alf, ul, up, xa, xb, &k, x2, &cx, wk);
```

Single precision:

```
ierr = ASL_rjtelg  (u, m, lt, ndiv, alf, ul, up, xa, xb, &k, x2, &cx, wk);
```

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	u	{D∗ / R∗}	m	Input	Random numbers
2	m	I	1	Input	Total number of random numbers
3	lt	I	1	Input	Test iteration count (See Note (a))
4	ndiv	I	1	Input	Number of subdivisions
5	alf	{D / R}	1	Input	Significance level (%)
6	ul	{D / R}	1	Input	Test interval lower limit (See Note (b))
7	up	{D / R}	1	Input	Test interval upper limit (See Note (b))
8	xa	{D / R}	1	Input	Mean value
9	xb	{D / R}	1	Input	Value of parameter β (See Note (c))
10	k	I*	1	Output	Number of passed tests
11	x2	{D∗ / R∗}	lt	Output	Test result χ^2 value
12	cx	{D∗ / R∗}	1	Output	χ^2 value for significance level

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
13	wk	$\left\{\begin{matrix}\text{D}* \\ \text{R}*\end{matrix}\right\}$	See Contents	Work	Work area **Size**: $2 \times \text{ndiv} + 4$
14	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) $\text{m} \geq \text{lt}$

(b) $\text{lt} \geq 1$

(c) $\text{ndiv} \geq 2$

(d) $\text{up} > \text{ul}$

(e) $0.0 < \text{alf} < 100.0$

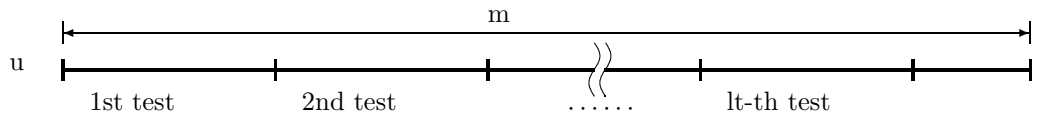
(f) $\text{xb} > 0.0$

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
1000	ul was too small, up was too large, or m was too small. (See Note (b))	Processing continues. (Test precision gets worse.)
3000	Restriction (a) was not satisfied.	Processing is aborted.
3010	Restriction (b) was not satisfied.	
3020	Restriction (c) was not satisfied.	
3030	Restriction (d) was not satisfied.	
3040	Restriction (e) was not satisfied.	
3050	Restriction (f) was not satisfied.	

(6) **Notes**

(a) The following figure shows the relationship between the total number of random numbers m, the test iteration count lt, and the random numbers u used for each test.



$\lfloor \text{m/lt} \rfloor$ random numbers u are taken at a time for each test, where $\lfloor x \rfloor$ represents the maximum integer that does not exceed x.

(b) If the test range lower limit ul is too small or if the upper limit up is too large, then an extremely small expected frequency range is tested and test precision worsens.
If F_{Ti} is the expected frequency in each partial interval, then ierr = 1000 if the following condition occurs in the function:

$$\text{F}_{\text{Ti}} < 5 \qquad (\text{i} = 1, \cdots, \text{ndiv})$$

Expressions for calculating ul and up, which are used as test criteria, are shown below.

$$(\mathrm{up}-\mathrm{ul})\times\min\left\{\frac{\exp(-\frac{\mathrm{ul}-\mathrm{xa}}{\mathrm{xb}})}{\left\{1+\exp(-\frac{\mathrm{ul}-\mathrm{xa}}{\mathrm{xb}})\right\}^2},\frac{\exp(-\frac{\mathrm{up}-\mathrm{xa}}{\mathrm{xb}})}{\left\{1+\exp(-\frac{\mathrm{up}-\mathrm{xa}}{\mathrm{xb}})\right\}^2}\right\}\geq\frac{5\times\mathrm{ndiv}\times\mathrm{xb}}{\left\lfloor\frac{\mathrm{m}}{\mathrm{lt}}\right\rfloor}$$

$\lfloor x\rfloor$ represents the maximum integer that does not exceed x.

(c) Variance is defined as follows.

$\sigma^2=\dfrac{\pi^2\beta^2}{3}$

(7) **Example**

(a) Problem

Perform the test 10 times with 10 subdivisions on 10000 logistic distribution random numbers having mean 0.0 and $\beta=1.0$.

(b) Input data

m = 10000, l = 10, ndiv = 10, alf = 5.0, ul = −4.7, up = 4.7, xa = 1.0 and xb = 1.0.

(c) Main program

```
/*      C interface example for ASL_djtelg */

#include <stdio.h>
#include <stdlib.h>
#include <asl.h>

int main()
{
    double *u;
    int m=10000;
    int l=10;
    int n=10;
    double alf=5.0;
    double ul= -4.7;
    double up=4.7;
    double xa=1.0;
    double xb=1.0;
    int k;
    double *x2;
    double cx;
    double *wk;
    int ierr;
    int i;

    int ix=1;
    int iy=1;

    printf( "    *** ASL_djtelg ***\n" );
    printf( "\n     ** Input **\n\n" );

    u = ( double * )malloc((size_t)( sizeof(double) * m ));
    if( u == NULL )
    {
        printf( "no enough memory for array u\n" );
        return -1;
    }

    x2 = ( double * )malloc((size_t)( sizeof(double) * l ));
    if( x2 == NULL )
    {
        printf( "no enough memory for array x2\n" );
        return -1;
    }

    wk = ( double * )malloc((size_t)( sizeof(double) * (2*n+4) ));
    if( wk == NULL )
    {
        printf( "no enough memory for array wk\n" );
        return -1;
    }

    ierr = ASL_djdblg(m, xa, xb, &ix, &iy, u);

    printf( "\t m = %8d\t\t  l = %8d\n", m, l );
    printf( "\t n = %8d\t\talf = %8.3g\n", n, alf );
    printf( "\tul = %8.3g\t\t up = %8.3g\n", ul, up );
    printf( "\txa = %8.3g\t\t xb = %8.3g\n", xa, xb );

    ierr = ASL_djtelg(u, m, l, n, alf, ul, up, xa, xb, &k, x2, &cx, wk);

    printf( "\n     ** Output **\n\n" );
```

```
        printf( "\tierr = %6d\n", ierr );

        printf( "\n\tNumber of Passed Test (k) = %6d\n", k );
        printf( "\n\t  Test No.       1       2       3       4       5" );
        printf(             "       6       7       8       9      10\n" );
        printf( "\tChi-Square\n" );
        printf( "\tValue (x2)" );
        for( i=0 ; i<l ; i++ )
        {
            printf( "%8.3g", x2[i] );
        }
        printf( "\n" );
        printf( "\n\tChi-Square Value for Percent Point (cx) = %8.3g\n", cx );

        free( u );
        free( x2 );
        free( wk );

        return 0;
    }
```

(d) Output results

```
*** ASL_djtelg ***

 ** Input **

 m =    10000         l =       10
 n =       10       alf =        5
ul =     -4.7        up =      4.7
xa =        1        xb =        1

 ** Output **

ierr =      0

Number of Passed Test (k) =     10

  Test No.       1       2       3       4       5       6       7       8       9      10
Chi-Square
Value (x2)    4.78    7.64    7.35    6.73     4.1    10.7    9.76    6.28    6.11    8.13

Chi-Square Value for Percent Point (cx) =     16.9
```

2.4 DISCRETE DISTRIBUTION RANDOM NUMBER TESTS

2.4.1 ASL_rjtebi
Binomial Distribution Random Number Test

(1) **Function**

Performs a frequency one-dimensional test on binomial distribution random numbers.

(2) **Usage**

Double precision:

Nothing

Single precision:

ierr = ASL_rjtebi (nl, m, lt, iup, alf, mn, p, &k, x2, &cx, wk);

(3) **Arguments and Return Value**

R:Single precision real C:Single precision complex I: $\left\{\begin{array}{l}\text{int as for 32bit Integer}\\ \text{long as for 64bit Integer}\end{array}\right\}$

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	nl	I*	m	Input	Random numbers
2	m	I	1	Input	Total number of random numbers
3	lt	I	1	Input	Iteration count (See Note (a))
4	iup	I	1	Input	Test interval upper limit (See Note (b))
5	alf	R	1	Input	Significance (%)
6	mn	I	1	Input	Number of trials
7	p	R	1	Input	Success probability
8	k	I*	1	Output	Number of passed tests
9	x2	R*	lt	Output	Test result χ^2 value
10	cx	R*	1	Output	χ^2 value for significance level
11	wk	R*	See Contents	Work	Work area **Size**: $2 \times (\text{iup} + 1)$
12	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) $\text{m} \geq \text{lt}$

(b) $\text{lt} \geq 1$

(c) $0.0 < \text{alf} < 100.0$

(d) $\text{mn} \geq 1$

(e) $0.0 < \text{p} < 1.0$

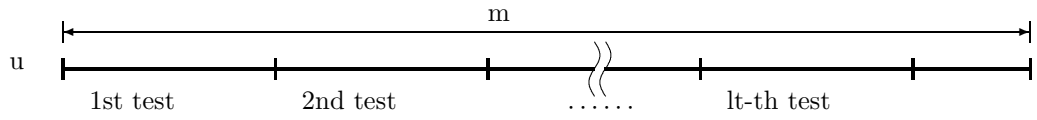
(f) $0 < \text{iup} \leq \text{mn}$

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
1000	iup was too large or m was too small. (See Note (b))	Processing continues. (Test precision gets worse.)
3000	Restriction (a), (b), (c), (d), (e) or (f) was not satisfied.	Processing is aborted.
4000	nl[i−1] < 0 or nl[i−1] > mn (i = 1, ··· , m)	

(6) **Notes**

(a) The following figure shows the relationship between the total number of random numbers m, the test iteration count lt, and the random numbers nl used for each test.



(b) If the test range upper limit iup is too large, then an extremely small expected frequency range is tested and test precision worsens.

If F_{Ti} is the expected frequency in each partial interval, then ierr = 1000 if the following condition occurs in the ASL_rjtebi function:

$$F_{Ti} < 5 \qquad (i = 1, \cdots, n)$$

The value of iup, which is used as a test criterion, is determined so that the following condition is satisfied.

$$\begin{pmatrix} mn \\ iup \end{pmatrix} p^{iup}(1-p)^{mn-iup} = \frac{5}{\lfloor m/lt \rfloor}$$

$\lfloor x \rfloor$ represents the maximum integer that does not exceed x.

Sample values of iup are shown below for p=0.5.

		⌊m/lt⌋			
		100	1, 000	10, 000	100, 000
mn	10	7	9	10	10
	50	29	33	36	38
	100	54	61	65	69

(7) **Example**

(a) Problem

Perform the test 10 times on 10000 binomial distribution random numbers with parmeters $m = 4$ and $p = 0.5$.

(b) Main program

```
/*      C interface example for ASL_rjtebi */

#include <stdio.h>
#include <stdlib.h>
#include <asl.h>

int main()
{
    int *nl;
    int m=10000;
    int l=10;
    int iup=4;
    float alf=1.0;
    int mn=4;
    float p=0.5;
    int k;
    float *x2;
    float cx;
    float *wk;
    int ierr;
    int i;

    int ix=1;
    int iy=1;
    int *iwk1;
    float *wk1;

    printf( "    *** ASL_rjtebi ***\n" );
    printf( "\n     ** Input **\n\n" );

    nl = ( int * )malloc((size_t)( sizeof(int) * m ));
    if( nl == NULL )
    {
        printf( "no enough memory for array nl\n" );
        return -1;
    }

    x2 = ( float * )malloc((size_t)( sizeof(float) * l ));
    if( x2 == NULL )
    {
        printf( "no enough memory for array x2\n" );
        return -1;
    }

    wk = ( float * )malloc((size_t)( sizeof(float) * (2*(iup+1)) ));
    if( wk == NULL )
    {
        printf( "no enough memory for array wk\n" );
        return -1;
    }

    iwk1 = ( int * )malloc((size_t)( sizeof(int) * (mn+2) ));
    if( iwk1 == NULL )
    {
        printf( "no enough memory for array iwk1\n" );
        return -1;
    }

    wk1 = ( float * )malloc((size_t)( sizeof(float) * (mn+2) ));
    if( wk1 == NULL )
    {
        printf( "no enough memory for array wk1\n" );
        return -1;
    }

    iwk1[0] = 0;
    wk1[0] = 0.0;


    ierr = ASL_rjdbbi(m, mn, p, &ix, &iy, nl, iwk1, wk1);

    printf( "\t  m = %6d\t\t  l =   %6d\n", m, l );
    printf( "\tiup = %6d\t\talf = %8.3g\n", iup, alf );
    printf( "\t mn = %6d\t\t  p = %8.3g\n", mn, p );

    ierr = ASL_rjtebi(nl, m, l, iup, alf, mn, p, &k, x2, &cx, wk);


    printf( "\n     ** Output **\n\n" );
    printf( "\tierr = %6d\n", ierr );

    printf( "\n\tNumber of Passed Test (k) = %6d\n", k );
    printf( "\n\t  Test No.       1       2       3       4       5" );
    printf(             "       6       7       8       9      10\n" );
    printf( "\tChi-Square\n" );
    printf( "\tValue (x2)" );
    for( i=0 ; i<l ; i++ )
    {
        printf( "%8.3g", x2[i] );
    }
```

```
        printf( "\n" );
        printf( "\n\tChi-Square Value for Percent Point (cx) = %8.3g\n", cx );

        free( nl );
        free( x2 );
        free( wk );
        free( iwk1 );
        free( wk1 );

        return 0;
    }
```

(c) Output results

```
        *** ASL_rjtebi ***

         ** Input **

          m =  10000          l =       10
        iup =      4        alf =        1
         mn =      4          p =      0.5

         ** Output **

        ierr =      0

        Number of Passed Test (k) =     10

          Test No.       1       2       3       4       5       6       7       8       9      10
        Chi-Square
        Value (x2)    4.57    9.62    2.48    1.89   0.936    1.18    3.76    2.41     2.2    2.54

        Chi-Square Value for Percent Point (cx) =     13.3
```

2.4.2 ASL_rjteng
Geometric Distribution Random Number Test

(1) **Function**

Performs a frequency one-dimensional test on geometric distribution random numbers.

(2) **Usage**

Double precision:

Nothing

Single precision:

ierr = ASL_rjteng (nl, m, lt, iup, alf, p, &k, x2, &cx, iwk);

(3) **Arguments and Return Value**

R:Single precision real C:Single precision complex I: $\left\{ \begin{array}{l} \text{int as for 32bit Integer} \\ \text{long as for 64bit Integer} \end{array} \right\}$

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	nl	I*	m	Input	Random numbers
2	m	I	1	Input	Total number of random numbers
3	lt	I	1	Input	Iteration count (See Note (a))
4	iup	I	1	Input	Test interval upper limit (See Note (b))
5	alf	R	1	Input	Significance (%)
6	p	R	1	Input	Success probability
7	k	I*	1	Output	Number of passed tests
8	x2	R*	lt	Output	Test result χ^2 value
9	cx	R*	1	Output	χ^2 value for significance level
10	iwk	I*	$2 \times$ (iup)	Work	Work area
11	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) $m \geq lt$

(b) $lt \geq 1$

(c) $0.0 < alf < 100.0$

(d) $0.0 < p < 1.0$

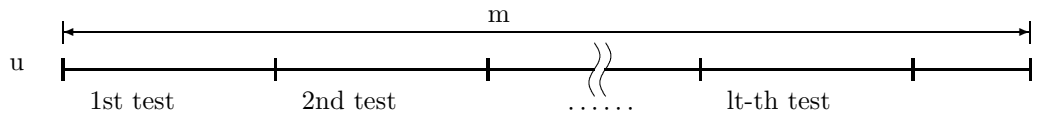
(e) $0 < iup$

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
1000	iup was too large or m was too small. (See Note (b))	Processing continues. (Test precision gets worse.)
3000	Restriction (a) was not satisfied.	Processing is aborted.
3010	Restriction (b) was not satisfied.	
3020	Restriction (c) was not satisfied.	
3030	Restriction (d) was not satisfied.	
3040	Restriction (e) was not satisfied.	

(6) **Notes**

(a) The following figure shows the relationship between the total number of random numbers m, the test iteration count lt, and the random numbers nl used for each test.



(b) If the test range upper limit iup is too large, then an extremely small expected frequency range is tested and test precision worsens.

If F_{Ti} is the expected frequency in each partial interval, then ierr = 1000 if the following condition occurs in the ASL_rjteng function:

$$F_{Ti} < 5 \qquad (i = 1, \cdots, n)$$

The value of iup, which is used as a test criterion, is determined so that the following condition is satisfied.

$$(1 - P)^{IUP-1} P = \frac{5}{\lfloor m/lt \rfloor}$$

$\lfloor x \rfloor$ represents the maximum integer that does not exceed x.

Sample values of iup are shown below for p=0.5.

$\lfloor m/lt \rfloor$	100	1000	10000	100000	1000000
iup	4	7	10	14	17

(7) **Example**

(a) Problem

Perform the test 10 times on 10000 geometric distribution random numbers with parameters $m = 4$ and $p = 0.5$.

(b) Main program

```
/*      C interface example for ASL_rjteng */

#include <stdio.h>
#include <stdlib.h>
#include <asl.h>

int main()
{
    int *nl;
    int m=10000;
    int l=10;
    int iup=6;
    float alf=1.0e0;
    float p=0.6e0;
    int k;
    float *x2;
    float cx;
    int *iwk;
    int ierr;

    int i;
    int ix=1;
    int iy=1;

    printf( "    *** ASL_rjteng ***\n" );
    printf( "\n     ** Input **\n\n" );

    nl = ( int * )malloc((size_t)( sizeof(int) * m ));
    if( nl == NULL )
    {
        printf( "no enough memory for array nl\n" );
        return -1;
    }

    x2 = ( float * )malloc((size_t)( sizeof(float) * l ));
    if( x2 == NULL )
    {
        printf( "no enough memory for array x2\n" );
        return -1;
    }

    iwk = ( int * )malloc((size_t)( sizeof(int) * (iup*2) ));
    if( iwk == NULL )
    {
        printf( "no enough memory for array iwk\n" );
        return -1;
    }

    iwk[0] = 0;

    ierr = ASL_rjdbng(m, p, &ix, &iy, nl);

    printf( "\t  m = %6d\t\t  l =   %6d\n", m, l );
    printf( "\tiup = %6d\t\talf = %8.3g\n", iup, alf );
    printf( "\t  p = %8.3g\n", p );

    ierr = ASL_rjteng(nl, m, l, iup, alf, p, &k, x2, &cx, iwk);

    printf( "\n     ** Output **\n\n" );
    printf( "\tierr = %6d\n", ierr );

    printf( "\n\tNumber of Passed Test (k) = %6d\n", k );
    printf( "\n\t  Test No.       1       2       3       4       5" );
    printf(                "       6       7       8       9      10\n" );
    printf( "\tChi-Square\n" );
    printf( "\tValue (x2)" );
    for( i=0 ; i<l ; i++ )
    {
        printf( "%8.3g", x2[i] );
    }
    printf( "\n" );
    printf( "\n\tChi-Square Value for Percent Point (cx) = %8.3g\n", cx );

    free( nl );
    free( x2 );
    free( iwk );

    return 0;
}
```

(c) Output results

```
*** ASL_rjteng ***

 ** Input **

  m =  10000          l =       10
iup =      6        alf =        1
  p =    0.6

 ** Output **

ierr =      0

Number of Passed Test (k) =     10

  Test No.       1       2       3       4       5       6       7       8       9      10
Chi-Square
Value (x2)    8.37    6.86    7.84    5.59    2.55    6.14    4.69    2.19    4.37    5.25

Chi-Square Value for Percent Point (cx) =     15.1
```

2.4.3 ASL_rjtepo
Poisson Distribution Random Number Test

(1) **Function**

Performs a frequency one-dimensional test on Poisson distribution random numbers.

(2) **Usage**

Double precision:

Nothing

Single precision:

ierr = ASL_rjtepo (nl, m, lt, iup, alf, am, &k, x2, &cx, wk);

(3) **Arguments and Return Value**

R:Single precision real C:Single precision complex I: $\left\{\begin{array}{l}\text{int as for 32bit Integer}\\ \text{long as for 64bit Integer}\end{array}\right\}$

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	nl	I*	m	Input	Random numbers
2	m	I	1	Input	Total number of random numbers
3	lt	I	1	Input	Test iteration count (See Note (a))
4	iup	I	1	Input	Test interval upper limit (See Note (b))
5	alf	R	1	Input	Significance level (%)
6	am	R	1	Input	Mean value
7	k	I*	1	Output	Number of passed tests
8	x2	R*	lt	Output	Test result χ^2 value
9	cx	R*	1	Output	χ^2 value for significance level
10	wk	R*	See Contents	Work	Work area **Size**: $2 \times (\text{iup} + 1)$
11	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) $\text{m} \geq \text{lt}$

(b) $\text{lt} \geq 1$

(c) $0.0 < \text{alf} < 100.0$

(d) $0.0 < \text{am} < \log_e$ (maximum value expressed by the computer)

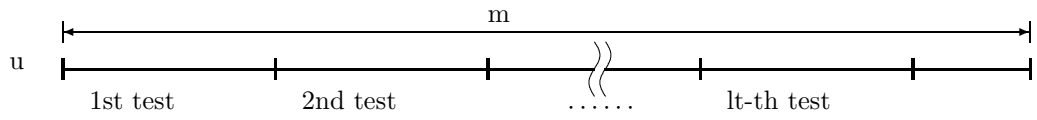
(e) $\text{iup} > 0$

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
1000	iup was too large or m was too small. (See Note (b))	Processing continues. (Test precision gets worse.)
3000	Restriction (a), (b), (c), (d) or (e) was not satisfied.	Processing is aborted.
4000	$\text{nl}[\text{i}-1] < 0 \quad (\text{i} = 1, \cdots, \text{m})$	

(6) **Notes**

(a) The following figure shows the relationship between the total number of random numbers m, the test iteration count lt, and the random numbers nl used for each test.



(b) If the test range upper limit iup is too large, then an extremely small expected frequency range is tested and test precision worsens.

If F_{Ti} is the expected frequency in each partial interval, then ierr = 1000 if the following condition occurs in the ASL_rjtepo function:

$$F_{Ti} < 5 \qquad (i = 1, \cdots, n)$$

The value of IUP, which is used as a test criterion, is determined so that the following condition is satisfied.

$$\frac{am^{iup}}{iup!} = \frac{5 \times e^{am}}{\lfloor m/lt \rfloor}$$

$\lfloor x \rfloor$ represents the maximum integer that does not exceed x.

Sample values of IUP are shown below for three different values of am.

		$\lfloor m/lt \rfloor$			
		100	1, 000	10, 000	100, 000
am	1.0	3	4	6	7
	5.0	8	11	13	15
	10.0	14	18	21	24

(7) **Example**

(a) Problem

Perform the test 10 times on 10000 Poisson distribution random numbers having mean value 1.0.

(b) Main program

```
/*      C interface example for ASL_rjtepo */

#include <stdio.h>
#include <stdlib.h>
#include <asl.h>

int main()
{
    int *nl;
    int m=10000;
    int l=10;
    int iup=4;
    float alf=1.0;
    float am=1.0;
    int k;
    float *x2;
    float cx;
    float *wk;
    int ierr;
    int i;

    int ix=1;
    int iy=1;
    int *iwk1;
    float *wk1;
```

```
    printf( "    *** ASL_rjtepo ***\n" );
    printf( "\n     ** Input **\n\n" );

    nl = ( int * )malloc((size_t)( sizeof(int) * m ));
    if( nl == NULL )
    {
        printf( "no enough memory for array nl\n" );
        return -1;
    }

    x2 = ( float * )malloc((size_t)( sizeof(float) * l ));
    if( x2 == NULL )
    {
        printf( "no enough memory for array x2\n" );
        return -1;
    }

    wk = ( float * )malloc((size_t)( sizeof(float) * (2*(iup+1)) ));
    if( wk == NULL )
    {
        printf( "no enough memory for array wk\n" );
        return -1;
    }

    iwk1 = ( int * )malloc((size_t)( sizeof(int) * (2*(int)am+12) ));
    if( iwk1 == NULL )
    {
        printf( "no enough memory for array iwk1\n" );
        return -1;
    }

    wk1 = ( float * )malloc((size_t)( sizeof(float) * (2) ));
    if( wk1 == NULL )
    {
        printf( "no enough memory for array wk1\n" );
        return -1;
    }

    iwk1[0] = 0;
    wk1[0] = 0.0;


    ierr = ASL_rjdbpo(m, am, &ix, &iy, nl, iwk1, wk1);

    printf( "\t  m =   %6d\t\t  l =   %6d\n", m, l );
    printf( "\tiup =   %6d\t\talf = %8.3g\n", iup, alf );
    printf( "\t am = %8.3g\n", am );

    ierr = ASL_rjtepo(nl, m, l, iup, alf, am, &k, x2, &cx, wk);


    printf( "\n     ** Output **\n\n" );
    printf( "\tierr = %6d\n", ierr );

    printf( "\n\tNumber of Passed Test (k) = %6d\n", k );
    printf( "\n\t  Test No.       1       2       3       4       5" );
    printf(                  "       6       7       8       9      10\n" );
    printf( "\tChi-Square\n" );
    printf( "\tValue (x2)" );
    for( i=0 ; i<l ; i++ )
    {
        printf( "%8.3g", x2[i] );
    }
    printf( "\n" );
    printf( "\n\tChi-Square Value for Percent Point (cx) = %8.3g\n", cx );

    free( nl );
    free( x2 );
    free( wk );
    free( iwk1 );
    free( wk1 );

    return 0;
}
```

(c) Output results

```
*** ASL_rjtepo ***

 ** Input **

  m =    10000        l =       10
iup =        4      alf =        1
 am =        1

 ** Output **

ierr =      0

Number of Passed Test (k) =     10
```

Test No.	1	2	3	4	5	6	7	8	9	10
Chi-Square Value (x2)	4.69	9.15	7.91	2.73	2.23	5.58	5.08	1.49	0.988	2.41

Chi-Square Value for Percent Point (cx) = 13.3

Chapter 3

PROBABILITY DISTRIBUTIONS

3.1 INTRODUCTION

In statistical analysis, processing for estimates or tests associate a variable called a random variable with various types of data. Random variables are broadly divided into **discrete random variables**, which correspond to cases in which the realized values are expressed by using discrete values such as $x_1, x_2, \cdots$ and **continuous random variables**, which correspond to cases in which the realized values take arbitrary values within a continuous interval such as $0 < x < 1$. With a discrete random variable, we can consider the probability ($Pr.\{X = x\}$) that the random variable (X) takes a specific value (x). However, with a continuous random variable, we consider the probability ($Pr.\{a \leq X < b\}$) that the random variable (X) takes a value within an arbitrary subinterval ($[a, b)$) in the interval where the realized values of the random variable (X) exist. Using the probability $Pr.\{x \leq X < x+dx\}$ that the random variable X takes an arbitrary value of the infinitesimal interval $[x, x + dx)$, the "function" $f(x)$ that satisfies

$$\int_x^{x+dx} f(u)du = Pr.\{x \leq X < x + dx\} \ (dx \to 0)$$

is called the probability density function (p.d.f.) of the continuous random variable X. From the definition of probability, we have

$$\int_{-\infty}^{\infty} f(u)du = 1$$

Normally, the value of $f(x)$ is set to zero for any interval in which the random variable X is not defined. The cumulative distribution function (c.d.f.) $F(x)$ is defined as the integral of the probability density function $f(x)$ as follows:

$$F(x) = \int_{-\infty}^{x} f(u)du$$

For practical use, the function $G(x)$, which is defined by the following expression, is also used as the cumulative distribution function.

$$G(x) = 1 - F(x) = \int_x^{\infty} f(u)du$$

These relationships can also easily be extended to several variables. For stricter definitions of the probability density function and cumulative distribution function or for unified handling of discrete and continuous random variables, refer to specialized technical texts. This library provides functions for computing the probability density function or cumulative distribution function of the following kinds of probability distributions as well as the values of their inverse functions.

- Normal Distribution
- Inverse of Normal Distribution
- Bivariate Normal Distribution
- χ^2 Distribution

- Inverse of χ^2 Distribution
- Noncentral χ^2 Distribution
- Inverse Noncentral χ^2 Distribution
- t Distribution
- Inverse of t Distribution
- Noncentral t Distribution
- Inverse Noncentral t Distribution
- F Distribution
- Inverse of F Distribution
- Gamma Distribution
- Inverse Gamma Distribution
- Beta Distribution
- Inverse Beta Distribution
- Uniform Distribution
- Triangular Distribution
- Pareto Distribution
- Weibull Distribution
- Exponential Distribution
- Gumbel Distribution
- Logarithmic Distribution
- Log-Normal Distribution
- Logistic Distribution
- Cauchy Distribution
- Binomial Distribution and Negative Binomial Distribution
- Geometric Distribution
- Poisson Distribution
- Hypergeometric Distribution
- Negative Hypergeometric Distribution

3.1.1 Explanation

(1) **Normal Distribution**
The probability density function of the normal distribution having mean μ and variance σ^2 is defined as follows.

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \quad (\sigma > 0)$$

(2) χ^2 **Distribution**
The probability density function of the χ^2 distribution having frequency χ^2 and number of degrees of freedom ν is defined as follows.

$$f(\chi^2|\nu) = \begin{cases} \frac{1}{2^{\frac{\nu}{2}}\Gamma(\frac{\nu}{2})}(\chi^2)^{\frac{\nu}{2}-1}e^{-\frac{\chi^2}{2}} & (\chi^2 > 0) \\ 0 & (\chi^2 \leq 0) \end{cases}$$

The mean and variance of the χ^2 distribution are given by the following equations.

$$E[\chi^2(\nu)] = \nu, \sigma^2[\chi^2(\nu)] = 2\nu$$

(3) **Noncentral** χ^2 **Distribution**
The probability density function of the noncentral χ^2 distribution having frequency χ^2, number of degrees of freedom ν, and noncentrality parameter λ is defined as follows.

$$f(x|\nu,\lambda) = \begin{cases} \frac{e^{-\frac{(x+\lambda)}{2}}x^{\frac{(\nu-2)}{2}}}{2^{\frac{\nu}{2}}} \sum_{k=0}^{\infty} \frac{\lambda^k x^k}{2^{2k}k!\Gamma(\frac{\nu}{2}+k)} & (x > 0) \\ 0 & (x \leq 0) \end{cases}$$

(4) t **Distribution**
The probability density function of the t distribution having frequency t and number of degrees of freedom ν is defined as follows.

$$f(t|\nu) = \frac{\Gamma(\frac{\nu+1}{2})}{\sqrt{\nu\pi}\Gamma(\frac{\nu}{2})(1+\frac{t^2}{\nu})^{\frac{\nu+1}{2}}}$$

The mean and variance of the t distribution are given by the following equations.

$$E[t(\nu)] = 0, \sigma^2[t(\nu)] = \frac{\nu}{\nu-2} \ (\nu > 2)$$

(5) **Noncentral** t **Distribution** The probability density function of the noncentral t distribution having frequency t, number of degrees of freedom ν, and noncentrality parameter δ is defined as follows.

$$f(t|\nu,\delta) = \frac{\nu^{\frac{\nu}{2}}e^{-\frac{\delta^2}{2}}}{\sqrt{\pi}\Gamma(\frac{\nu}{2})(\nu+t^2)^{\frac{(\nu+1)}{2}}} \sum_{k=0}^{\infty} \Gamma(\frac{\nu+k+1}{2}) \frac{\delta^k}{k!} (\frac{2t^2}{\nu+t^2})^{\frac{k}{2}}$$

(6) F **Distribution**

The probability density function of the F distribution having frequency F and numbers of degrees of freedom ν_1 and ν_2 is defined as follows.

$$f(x|\nu_1,\nu_2)=\frac{\nu_1^{\frac{\nu_1}{2}}\cdot\nu_2^{\frac{\nu_2}{2}}\cdot x^{\frac{\nu_1}{2}-1}}{B(\frac{\nu_1}{2},\frac{\nu_2}{2})(\nu_1x+\nu_2)^{\frac{\nu_1+\nu_2}{2}}}=\frac{1}{B(\frac{\nu_1}{2},\frac{\nu_2}{2})}(\frac{\nu_1}{\nu_2})^{\frac{\nu_1}{2}}(1+\frac{\nu_1}{\nu_2}x)^{-\frac{\nu_1+\nu_2}{2}}x^{\frac{\nu_1}{2}-1}$$

The mean and variance of the F distribution are given by the following equations.

$$E[F]=\frac{\nu_2}{\nu_2-2}\ (\nu_2>2),\sigma^2[F]=\frac{2\nu_2^2(\nu_1+\nu_2-2)}{\nu_1(\nu_2-2)^2(\nu_2-4)}\ (\nu_2>4)$$

(7) **Gamma Distribution**

The probability density function of the gamma distribution having parameters α and β is defined as follows.

$$f(x;\alpha,\beta)=\begin{cases}\frac{\beta^\alpha}{\Gamma(\alpha)}x^{\alpha-1}e^{-\beta x} & (x>0;\alpha,\beta>0)\\ 0 & (x\le 0;\alpha,\beta>0)\end{cases}$$

The mean and variance of the gamma distribution are given by the following equations.

$$E[x]=\frac{\alpha}{\beta},\sigma^2[x]=\frac{\alpha}{\beta^2}$$

(8) **Beta Distribution**

The probability density function of the beta distribution having parameters a and b is defined as follows.

$$f(x;a,b)=\begin{cases}\frac{1}{B(a,b)}x^{a-1}(1-x)^{b-1} & (0<x<1;a,b>0)\\ 0 & (x\le 0,x\ge 1;a,b>0)\end{cases}$$

(9) **Uniform Distribution**

The probability density function of the uniform distribution within the interval (a,b) is defined as follows.

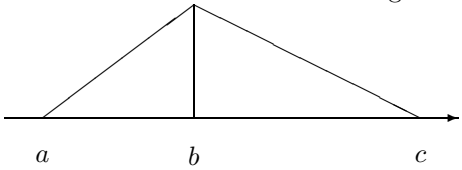
$$f(x)=\begin{cases}\frac{1}{b-a} & (a\le x\le b)\\ 0 & (x<a,x>b)\end{cases}$$

The mean and variance of the uniform distribution are given by the following equations.

$$E[x]=\frac{a+b}{2},\sigma^2[x]=\frac{(b-a)^2}{12}$$

(10) **Triangular Distribution**

Take the coordinates of the triangle as shown in the following figure.



a:x coordinate of the left end of the triangular distribution

b:x coordinate of the apex of the triangular distribution

c:x coordinate of the right end of the triangular distribution
The probability density function of the triangular distribution at this time is defined as follows.

$$f(x) = \begin{cases} \dfrac{2(x-a)}{(b-a)(c-a)} & (a \leq x \leq b) \\ \dfrac{2(c-x)}{(c-a)(c-b)} & (b < x \leq c) \\ 0 & (x < a, x > c) \end{cases}$$

(11) **Pareto Distribution**
The probability density function of the Pareto distribution having parameters a and b ($a > 1$, $b > 0$) is defined as follows.

$$f(x;a,b) = \begin{cases} (a-1)(\dfrac{x}{b})^{-a}\dfrac{1}{b} & (x > b; a > 1, b > 0) \\ 0 & (x \leq b; a > 1, b > 0) \end{cases}$$

(12) **Weibull Distribution**
The probability density function of the Weibull distribution having parameters a and b ($a > 0$, $b > 0$) is defined as follows.

$$f(x;a,b) = \begin{cases} a\left(\dfrac{x}{b}\right)^{a-1} e^{-(\frac{x}{b})^a}\dfrac{1}{b} & (0 < x; a, b > 0) \\ 0 & (x \leq 0; a, b > 0) \end{cases}$$

(13) **Exponential Distribution**
A gamma distribution in which the parameter α is 1 is called the exponential distribution. In the exponential distribution, λ is used in place of the parameter β. The probability density function is defined as follows.

$$f(x;\lambda) = \begin{cases} \lambda e^{-\lambda x} & (x > 0; \lambda > 0) \\ 0 & (x \leq 0; \lambda > 0) \end{cases}$$

The mean and variance of the exponential distribution are given by the following equations.

$$E[x] = \frac{1}{\lambda}, \sigma^2[x] = \frac{1}{\lambda^2}$$

(14) **Gumbel Distribution**
The probability density function of the Gumbel distribution having parameters a and b ($b > 0$) is defined as follows.

$$f(x;a,b) = \frac{1}{b} e^{\frac{x-a}{b}} e^{-e^{\frac{x-a}{b}}}$$

(15) **Logarithmic Distribution**
The probability density function of the Logarithmic distribution parameters within the interval (a, b) is defined as follows.

$$f(x;a,b) = \frac{\log x}{b(\log b - 1) - a(\log a - 1)}$$

(16) **Log-Normal Distribution**
The probability density function of the log-normal distribution having mean $e^{\mu}\sqrt{e^{\sigma^2}}$ and variance $e^{2\mu}e^{\sigma^2}(e^{\sigma^2} - 1)$ is defined as follows.

$$f(x) = \frac{1}{x\sigma\sqrt{2\pi}} e^{-\frac{(\ln x - \mu)^2}{2\sigma^2}} \quad (\sigma > 0)$$

(17) **Logistic Distribution**

The probability density function of the logistic distribution having mean α and variance σ^2 is defined as follows.

$$\begin{aligned} &f(x) = \frac{e^{-\frac{x-\alpha}{\beta}}}{\beta\left\{1+e^{-\frac{x-\alpha}{\beta}}\right\}^2} \\ &(-\infty < x < \infty, -\infty < \alpha < \infty, \beta > 0) \\ &\sigma^2 = \frac{\pi^2\beta^2}{3} \end{aligned} \tag{3.1}$$

(18) **Cauchy Distribution**

The probability density function of Cauchy distribution having parameters α and β is defined as follows.

$$f(x;\alpha,\beta) = \frac{1}{\pi}\left[\frac{\beta}{\beta^2+(x-\alpha)^2}\right] \quad (\beta > 0)$$

(19) **Binomial Distribution and Negative Binomial Distribution**

Given the probability that an event will occur p, the number of trials n, and the number of occurrences m, the binomial distribution probability in m occurrences is defined as follows.

$$P_{BIN}(X = m; p, n) = \left(\begin{array}{c} n \\ m \end{array}\right) p^m \cdot q^{n-m} \quad (q = 1-p)$$

The mean and variance of the binomial distribution are given by the following equations.

$$E[m] = np, \sigma^2[m] = np(1-p)$$

Given the probability of success in one trial p and the number of successes n and number of failures m in repeated trials, the negative binomial distribution probability in m failures is defined as follows.

$$P_{NB}(X = m; p, n) = \left(\begin{array}{c} n+m-1 \\ m \end{array}\right) p^n \cdot q^m \quad (q = 1-p)$$

The mean and variance of the negative binomial distribution are given by the following equations.

$$E[m] = \frac{n}{p}, \sigma^2[m] = \frac{n(1-p)}{p^2}$$

(20) **Geometric Distribution**

Given the probability of success in m trial p, the ASL_d1ddbp or ASL_r1ddbp obtains the values of the geometric distribution probability $P_{NB}(X = m; p)$ is defined by the following equations.

$$P_{NB}(X = m; p) = q^{m-1}p \qquad (q = 1-p)$$

The mean and variance of the Geometric distribution are given by the following equations.

$$E[m] = \frac{1}{p}, \sigma^2[m] = \frac{q}{p^2}$$

(21) **Poisson Distribution**

Given the mean λ and random variable k, the value of the probability $Pr.\{X = k\}$ of a Poisson distribution is defined by the following expressions.

$$Pr.\{X = k\} = e^{-\lambda}\frac{\lambda^k}{k!} \qquad (k = 0, 1, 2, \cdots; \lambda > 0)$$

(22) **Hypergeometric Distribution**

Assume that there is a lot of size N in which M of the N articles are inferior goods and $N - M$ articles are of good quality. When an arbitrary sample of size n is extracted from this lot, the hypergeometric distribution probability $Pr.\{X = k\}$ corresponding to the probability distribution in which k inferior goods appear is defined as follows.

$$Pr.\{X = k\} = \begin{cases} \dfrac{\begin{pmatrix} M \\ k \end{pmatrix}\begin{pmatrix} N-M \\ n-k \end{pmatrix}}{\begin{pmatrix} N \\ n \end{pmatrix}} & k = 0, 1, 2, \cdots, \min\{M, n\} \\ 0 & Otherwise \end{cases}$$

The mathematical expectation and variance of the hypergeometric distribution are given by the following equations.

$$E(X) = np, \sigma^2(X) = \frac{N-k}{N-1}np(1-p) \quad (p = \frac{M}{N})$$

(23) **Negative Hypergeometric Distribution**

Assume that there is a lot of size NN in which M of the NN articles are inferior goods and $NN - M$ articles are of good quality. Sampling from this lot is continued until n inferior goods are extracted. The negative hypergeometric distribution probability $Pr.\{X = k\}$ is defined as the probability of such occurrences that exactly k goods has been extracted at this time. The probability probability $Pr.\{X = k\}$ is defined as follows.

$$Pr.\{X = k\} = \begin{cases} \dfrac{\begin{pmatrix} M \\ NN-1 \end{pmatrix}\begin{pmatrix} NN-M \\ k-n \end{pmatrix}}{\begin{pmatrix} NN \\ k-1 \end{pmatrix}} \times \dfrac{M-n+1}{NN-k+1} \\ = \dfrac{\begin{pmatrix} k-1 \\ n-1 \end{pmatrix}\begin{pmatrix} NN-k \\ M-n \end{pmatrix}}{\begin{pmatrix} NN \\ M \end{pmatrix}} & \text{When} k = n, n+1, n+2, \cdots, NN-M+n \\ 0 & \text{Otherwise.} \end{cases}$$

$$F(k) = \sum_{i=n}^{k} Pr.\{X = i\} = \frac{\sum_{i=0}^{k}\begin{pmatrix} i-1 \\ n-1 \end{pmatrix}\begin{pmatrix} NN-i \\ M-n \end{pmatrix}}{\begin{pmatrix} NN \\ M \end{pmatrix}}$$

3.1.2 Reference Bibliography

(1) Feller, W. , "An introduction to probability theory and its applications: I (3rd ed.) II (2nd ed.)", John Wiley & Sons, New York (1968, 1970)

(2) Abramowitz, M. and Stegur, I. A. , eds. , "Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables", Dover Publications, Inc. (1965).

3.2 CONTINUOUS DISTRIBUTIONS

3.2.1 ASL_d1cdno, ASL_r1cdno Normal Distribution

(1) **Function**

For a normal distribution having mean μ and variance σ^2, the ASL_d1cdno or ASL_r1cdno obtains the values of the following functions.

(a) probability density function; p.d.f.

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \quad (\sigma > 0)$$

(b) cumulative distribution function; c.d.f.

$$P(x) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(t-\mu)^2}{2\sigma^2}} dt \quad (\sigma > 0)$$

(c) c.d.f.

$$Q(x) = 1 - P(x) = \int_{x}^{\infty} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(t-\mu)^2}{2\sigma^2}} dt \quad (\sigma > 0)$$

(2) **Usage**

Double precision:

```
ierr = ASL_d1cdno  (xe, xv, xi, &xo, isw);
```

Single precision:

```
ierr = ASL_r1cdno  (xe, xv, xi, &xo, isw);
```

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	xe	{D / R}	1	Input	Value of mean μ
2	xv	{D / R}	1	Input	Value of variance σ^2
3	xi	{D / R}	1	Input	Value of random variable x
4	xo	{D∗ / R∗}	1	Output	Value of the probability density function $f(x)$ of the normal distribution or of the cumulative distribution function $P(x)$ or $Q(x)$ of the normal distribution.
5	isw	I	1	Input	isw=0: Obtain the value of the probability density function $f(x)$ for xo isw=1: Obtain the value of the cumulative distribution function $P(x)$ for xo isw=2:Obtain the value of the cumulative distribution function $Q(x)$ for xo
6	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) isw $\in \{0, 1, 2\}$

(b) xv > 0.0

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
3000	Restriction (a) was not satisfied.	Processing is aborted.
3010	Restriction (b) was not satisfied.	

(6) **Notes**

(a) From the relational expression $P(x) + Q(x) = 1$, it is possible to obtain either $P(x)$ or $Q(x)$ from the other. However, there may be times when cancellation of significant digits occurs, preventing good precision from being obtained.

(b) A random variable that obeys a normal distribution having parameters μ and σ is usually represented by $N(\mu, \sigma^2)$. When $\mu = 0$ and $\sigma = 1$, the random variable $N(0, 1)$ is called the standard random variable, and the probability distribution is called the standard normal distribution.

(7) **Example**

(a) Problem

Let $\mu = 5.0$, $\sigma^2 = 2.5$ and $x = 3.0$ and obtain the values of the probability density function $f(x)$ and the cumulative distribution functions $P(x)$ and $Q(x)$.

(b) Input data

xe = 5.0, xv = 2.5 and xi = 3.0.

(c) Main program

```
/*      C interface example for ASL_d1cdno */

#include <stdio.h>
#include <asl.h>

int main()
{
    double xe;
    double xv;
    double xi;
    double xo;
    int isw;
    int ierr;

    xe=5.0;
    xv=2.5;
    xi=3.0;

    printf( "    *** ASL_d1cdno ***\n" );
    printf( "\n     ** Input **\n\n" );

    printf( "\txe = %8.3g xv = %8.3g xi = %8.3g\n", xe, xv, xi );

    printf( "\n     ** Output **\n\n" );
    isw=0;
    ierr = ASL_d1cdno(xe, xv, xi, &xo, isw);

    printf( "\tierr = %6d\n", ierr );
    printf( "\tValue of P.D.F   =%8.3g\n\n", xo );
    isw=1;
    ierr = ASL_d1cdno(xe, xv, xi, &xo, isw);

    printf( "\tierr = %6d\n", ierr );
    printf( "\tValue of C.D.F(1)=%8.3g\n\n", xo );
    isw=2;
    ierr = ASL_d1cdno(xe, xv, xi, &xo, isw);

    printf( "\tierr = %6d\n", ierr );
    printf( "\tValue of C.D.F(2)=%8.3g\n\n", xo );

    return 0;
}
```

(d) Output results

```
*** ASL_d1cdno ***

 ** Input **

xe =        5 xv =      2.5 xi =        3

 ** Output **

ierr =      0
Value of P.D.F   =   0.113

ierr =      0
Value of C.D.F(1)=   0.103

ierr =      0
Value of C.D.F(2)=   0.897
```

3.2.2 ASL_d1cdin, ASL_r1cdin
Inverse of Normal Distribution

(1) **Function**

When given the cumulative distribution function (c.d.f.) $P(x)$ or $Q(x)$ of the normal distribution having mean μ and variance σ^2, the ASL_d1cdin or ASL_r1cdin obtains the random variable value x at that time. $P(x)$ and $Q(x)$ are defined by the following equations.

$$P(x) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(t-\mu)^2}{2\sigma^2}} dt \quad (\sigma > 0)$$

$$Q(x) = 1 - P(x) = \int_{x}^{\infty} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(t-\mu)^2}{2\sigma^2}} dt \quad (\sigma > 0)$$

(2) **Usage**

Double precision:

ierr = ASL_d1cdin (xe, xv, xi, &xo, isw);

Single precision:

ierr = ASL_r1cdin (xe, xv, xi, &xo, isw);

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	xe	{D / R}	1	Input	Value of mean μ
2	xv	{D / R}	1	Input	Value of variance σ^2
3	xi	{D / R}	1	Input	Value of the cumulative distribution function $P(x)$ or $Q(x)$ of the normal distribution.
4	xo	{D∗ / R∗}	1	Output	Value of random variable x
5	isw	I	1	Input	isw=1: Input the value of $P(x)$ for xi isw=2: Input the value of $Q(x)$ for xi
6	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) isw $\in \{1, 2\}$

(b) xv > 0.0

(c) $0.0 \leq$ xi ≤ 1.0

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
1000	xi = 0.0 or xi = 1.0	The positive maximum value or negative minimum value is set for xo.
3000	Restriction (a) was not satisfied.	Processing is aborted.
3010	Restriction (b) was not satisfied.	
3020	Restriction (c) was not satisfied.	
3500	The maximum number of iterations was reached before the specified precision was obtained.	The value at that time is returned.

(6) **Notes**

(a) A random variable that obeys a normal distribution having parameters μ and σ is usually represented by $N(\mu, \sigma^2)$. When $\mu = 0$ and $\sigma = 1$, the random variable $N(0, 1)$ is called the standard random variable, and the probability distribution is called the standard normal distribution.

(7) **Example**

(a) Problem
For $\mu = 5.0$ and $\sigma^2 = 2.5$, obtain the values of x for which $P(x) = 0.2$ and $Q(x) = 0.2$ occur, respectively.

(b) Input data
xe = 5.0, xv = 2.5 and xi = 0.2.

(c) Main program

```
/*      C interface example for ASL_d1cdin */

#include <stdio.h>
#include <asl.h>

int main()
{
    double xe;
    double xv;
    double xi;
    double xo;
    int isw;
    int ierr;


    xe=5.0;
    xv=2.5;
    xi=0.2;

    printf( "    *** ASL_d1cdin ***\n" );
    printf( "\n     ** Input **\n\n" );

    printf( "\txe = %8.3g xv = %8.3g xi = %8.3g\n", xe, xv, xi );


    printf( "\n     ** Output **\n\n" );
    isw=1;
    ierr = ASL_d1cdin(xe, xv, xi, &xo, isw);

    printf( "\tierr = %6d\n", ierr );
    printf( "\tValue x Corresponding to P(x)=%8.3g\n\n", xo );
    isw=2;
    ierr = ASL_d1cdin(xe, xv, xi, &xo, isw);

    printf( "\tierr = %6d\n", ierr );
    printf( "\tValue x Corresponding to Q(x)=%8.3g\n\n", xo );


    return 0;
}
```

(d) Output results

```
*** ASL_d1cdin ***

 ** Input **

xe =        5 xv =      2.5 xi =      0.2

 ** Output **

ierr =      0
Value x Corresponding to P(x)=    3.67

ierr =      0
Value x Corresponding to Q(x)=    6.33
```

3.2.3 ASL_d1cdbn, ASL_r1cdbn Bivariate Normal Distribution

(1) **Function**

For a bivariate normal distribution having means μ_x, μ_y, variances ${\sigma_x}^2, {\sigma_y}^2$ and correlation coefficient ρ, the ASL_d1cdbn or ASL_r1cdbn obtains the values of the following functions.

(a) probability density function (p.d.f.)

$$f(x,y) = \frac{1}{2\pi\sigma_x\sigma_y\sqrt{1-\rho^2}} e^{-\frac{Q}{2(1-\rho^2)}} \quad (\sigma_x, \sigma_y > 0)$$

where,

$$Q = \frac{(x-\mu_x)^2}{{\sigma_x}^2} - \frac{2\rho(x-\mu_x)(y-\mu_y)}{\sigma_x\sigma_y} + \frac{(y-\mu_y)^2}{{\sigma_y}^2} \quad (\sigma_x, \sigma_y > 0)$$

(b) cumulative distribution function; c.d.f.

$$B(h,k;\rho) = \int_{-\infty}^{h}\int_{-\infty}^{k} f(x,y)\ dxdy$$

(c) c.d.f.

$$L(h,k;\rho) = \int_{h}^{\infty}\int_{k}^{\infty} f(x,y)\ dxdy$$

(2) **Usage**

Double precision:

```
ierr = ASL_d1cdbn  (xe, ye, xv, yv, xh, yk, rho, &xo, isw);
```

Single precision:

```
ierr = ASL_r1cdbn  (xe, ye, xv, yv, xh, yk, rho, &xo, isw);
```

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	xe	{D / R}	1	Input	Value of mean μ_x.
2	ye	{D / R}	1	Input	Value of mean μ_y.
3	xv	{D / R}	1	Input	Value of variance ${\sigma_x}^2$.
4	yv	{D / R}	1	Input	Value of variance ${\sigma_y}^2$.

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
5	xh	$\left\{\begin{matrix}\mathrm{D}\\ \mathrm{R}\end{matrix}\right\}$	1	Input	isw=0:Value of random variable x isw=1:Upper bound h of the integration range of random variable x isw=2:Lower bound h of the integration range of random variable x
6	yk	$\left\{\begin{matrix}\mathrm{D}\\ \mathrm{R}\end{matrix}\right\}$	1	Input	isw=0:Value of random variable y isw=1:Upper bound k of the integration range of random variable y isw=2:Lower bound k of the integration range of random variable y
7	rho	$\left\{\begin{matrix}\mathrm{D}\\ \mathrm{R}\end{matrix}\right\}$	1	Input	Value of correlation coefficient ρ.
8	xo	$\left\{\begin{matrix}\mathrm{D}*\\ \mathrm{R}*\end{matrix}\right\}$	1	Output	Value of the probability density function $f(x, y)$ or of the cumulative distribution function $B(h, k; \rho)$ or $L(h, k; \rho)$ of the bivariate normal distribution.
9	isw	I	1	Input	Processing switch isw=0: Obtain the value of the probability density function $f(x, y)$ for xo isw=1:Obtain the value of the cumulative distribution function $B(h, k; \rho)$ for xo isw=2:Obtain the value of the cumulative distribution function $L(h, k; \rho)$ for xo
10	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) isw $\in \{0, 1, 2\}$

(b) xv, yv > 0.0

(c) $-1.0 \leq \rho \leq 1.0$

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
3000	Restriction (a) was not satisfied.	Processing is aborted.
3010	Restriction (b) was not satisfied.	
3020	Restriction (c) was not satisfied.	
3500	The maximum number of iterations was reached before the specified precision was obtained.	The value at that time is returned.

(6) **Notes**

None

(7) **Example**

(a) Problem

Let $\mu_x = 1.0, \mu_y = 2.0, {\sigma_x}^2 = 3.0, {\sigma_y}^2 = 4.0,\ x = h = 5.0, y = k = 6.0$ and $\rho = 0.7$ and obtain the values of the probability density function $f(x, y)$ and the cumulative distribution functions $B(h, k; \rho)$ and $L(h, k; \rho)$.

(b) Input data

xe $= 1.0,$ ye $= 2.0,$ xv $= 3.0,$ yv $= 4.0,$ xh $= 5.0,$ yk $= 6.0$ and rho $= 0.7$.

(c) Main program

```
/*      C interface example for ASL_d1cdbn */

#include <stdio.h>
#include <stdlib.h>
#include <asl.h>

int main()
{
    double xe;
    double ye;
    double xv;
    double yv;
    double xhin;
    double ykin;
    double rho;
    double xo;
    int isw;
    int ierr;


    printf( "    *** ASL_d1cdbn ***\n" );
    printf( "\n     ** Input **\n\n" );

    xe = 1.0;
    ye = 2.0;
    xv = 3.0;
    yv = 4.0;
    xhin = 5.0;
    ykin = 6.0;
    rho = 0.7;

    printf( "\txe  = %8.3g\n", xe );
    printf( "\tye  = %8.3g\n", ye );
    printf( "\txv  = %8.3g\n", xv );
    printf( "\tyv  = %8.3g\n", yv );
    printf( "\txh  = %8.3g\n", xhin );
    printf( "\tyk  = %8.3g\n", ykin );
    printf( "\trho = %8.3g\n", rho );

    isw = 0;
    ierr = ASL_d1cdbn(xe, ye, xv, yv, xhin, ykin, rho, &xo, isw);

    printf( "\n\n     ** Output **\n\n" );
    printf( "\t ierr = %6d\n\n", ierr );
    printf( "\t P.D.F = %8.3g\n\n", xo );
```

```
        isw = 1;
        ierr = ASL_d1cdbn(xe, ye, xv, yv, xhin, ykin, rho, &xo, isw);

        printf( "\n     ** Output **\n\n" );
        printf( "\t ierr = %6d\n\n", ierr );
        printf( "\t C.D.F(1) = %8.3g\n\n", xo );

        isw = 2;
        ierr = ASL_d1cdbn(xe, ye, xv, yv, xhin, ykin, rho, &xo, isw);

        printf( "\n     ** Output **\n\n" );
        printf( "\t ierr = %6d\n\n", ierr );
        printf( "\t C.D.F(2) = %8.3g\n\n", xo );

        return 0;
    }
```

(d) Output results

```
*** ASL_d1cdbn ***

 ** Input **

xe  =        1
ye  =        2
xv  =        3
yv  =        4
xh  =        5
yk  =        6
rho =      0.7


 ** Output **

 ierr =      0

 P.D.F =  0.00387


 ** Output **

 ierr =      0

 C.D.F(1) =    0.971


 ** Output **

 ierr =      0

 C.D.F(2) =   0.0044
```

3.2.4 ASL_d1cdch, ASL_r1cdch χ^2 Distribution

(1) **Function**

For a χ^2 distribution having frequency χ^2 and number of degrees of freedom ν, the ASL_d1cdch or ASL_r1cdch obtains the values of the following functions.

(a) probability density function; p.d.f.

$$f(\chi^2|\nu) = \begin{cases} \dfrac{1}{2^{\frac{\nu}{2}}\Gamma(\frac{\nu}{2})}(\chi^2)^{\frac{\nu}{2}-1}e^{-\frac{\chi^2}{2}} & (\chi^2 > 0) \\ 0 & (\chi^2 \leq 0) \end{cases}$$

(b) cumulative distribution function; c.d.f.

$$P(\chi^2|\nu) = \int_0^{\chi^2} \frac{1}{2^{\frac{\nu}{2}}\Gamma(\frac{\nu}{2})}(t)^{\frac{\nu}{2}-1}e^{-\frac{t}{2}}dt \quad (0 \leq \chi^2 < \infty)$$

(c) c.d.f.

$$Q(\chi^2|\nu) = 1 - P(\chi^2|\nu) = \int_{\chi^2}^{\infty} \frac{1}{2^{\frac{\nu}{2}}\Gamma(\frac{\nu}{2})}(t)^{\frac{\nu}{2}-1}e^{-\frac{t}{2}}dt \quad (0 \leq \chi^2 < \infty)$$

(2) **Usage**

Double precision:

```
ierr = ASL_d1cdch  (n, xi, &xo, isw);
```

Single precision:

```
ierr = ASL_r1cdch  (n, xi, &xo, isw);
```

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	n	I	1	Input	Value of number of degrees of freedom ν
2	xi	{D / R}	1	Input	Value of frequency χ^2
3	xo	{D∗ / R∗}	1	Output	Value of the probability density function $f(\chi^2\|\nu)$ of the χ^2 distribution or of the cumulative distribution function $P(\chi^2\|\nu)$ or $Q(\chi^2\|\nu)$ of the χ^2 distribution.
4	isw	I	1	Input	isw=0: Obtain the value of the probability density function $f(\chi^2\|\nu)$ for xo isw=1: Obtain the value of the cumulative distribution function $P(\chi^2\|\nu)$ for xo isw=2: Obtain the value of the cumulative distribution function $Q(\chi^2\|\nu)$ for xo
5	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) isw $\in \{0, 1, 2\}$

(b) n ≥ 1

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
1000	xi ≤ 0.0	0.0 or 1.0 is set for xo.
3000	Restriction (a) was not satisfied.	Processing is aborted.
3010	Restriction (b) was not satisfied.	

(6) **Notes**

(a) From the relational expression $P(\chi^2|\nu)+Q(\chi^2|\nu) = 1$, it is possible to obtain either $P(\chi^2|\nu)$ or $Q(\chi^2|\nu)$ from the other. However, there may be times when cancellation of significant digits occurs, preventing good precision from being obtained.

(b) The mean and variance of a χ^2 distribution having n degrees of freedom are given by the following equations.

$$E[\chi^2(n)] = n, \sigma^2[\chi^2(n)] = 2n$$

(c) If u is the random variable that obeys the standard normal distribution $N(0, 1)$, then the distribution of u^2 is the χ^2 distribution having 1 degree of freedom.

(d) If $X_i (i = 1, \cdots, n)$ are the random variables of an arbitrary sample of size n extracted from a normal population $(N(\mu, \sigma^2))$ having mean μ and variance σ^2, the following expressions

$$\frac{\sum_{i=1}^{n}(X_i - \overline{X})^2}{\sigma^2} \text{ and } \frac{n(\overline{X} - \mu)^2}{\sigma^2}$$

obey χ^2 distributions having $n - 1$ and 1 degrees of freedom respectively, and they are mutually independent.

(7) **Example**

(a) Problem

Let $\chi^2 = 5.0$ and $\nu = 2$ and obtain the values of the probability density function $f(\chi^2|\nu)$ and the cumulative distribution functions $P(\chi^2|\nu)$ and $Q(\chi^2|\nu)$.

(b) Input data

xi = 5.0 and n = 2.

(c) Main program

```
/*      C interface example for ASL_d1cdch */

#include <stdio.h>
#include <asl.h>

int main()
{
    int n;
    double xi;
    double xo;
    int isw;
    int ierr;

    n=2;
    xi=5.0;

    printf( "    *** ASL_d1cdch ***\n" );
    printf( "\n     ** Input **\n\n" );

    printf( "\tn = %6d xi = %8.3g\n", n, xi );

    printf( "\n     ** Output **\n\n" );
    isw=0;
    ierr = ASL_d1cdch(n, xi, &xo, isw);
    printf( "\tierr = %6d\n", ierr );

    printf( "\tValue of P.D.F   =%8.3g\n\n", xo );
    isw=1;
    ierr = ASL_d1cdch(n, xi, &xo, isw);
    printf( "\tierr = %6d\n", ierr );

    printf( "\tValue of C.D.F(1)=%8.3g\n\n", xo );
    isw=2;
    ierr = ASL_d1cdch(n, xi, &xo, isw);
    printf( "\tierr = %6d\n", ierr );

    printf( "\tValue of C.D.F(2)=%8.3g\n\n", xo );

    return 0;
}
```

(d) Output results

```
*** ASL_d1cdch ***

 ** Input **

n =      2 xi =        5

 ** Output **

ierr =      0
Value of P.D.F   =   0.041

ierr =      0
Value of C.D.F(1)=   0.918

ierr =      0
Value of C.D.F(2)=  0.0821
```

3.2.5 ASL_d1cdic, ASL_r1cdic Inverse of χ^2 Distribution

(1) **Function**

When given the cumulative distribution function (c.d.f.) $P(\chi^2|\nu)$ or $Q(\chi^2|\nu)$ of the χ^2 distribution for which the number of degrees of freedom is ν, the ASL_d1cdic or ASL_r1cdic obtains the frequency χ^2 at that time. $P(\chi^2|\nu)$ and $Q(\chi^2|\nu)$ are defined by the following equations.

$$P(\chi^2|\nu) = \int_0^{\chi^2} \frac{1}{2^{\frac{\nu}{2}}\Gamma(\frac{\nu}{2})}(t)^{\frac{\nu}{2}-1}e^{-\frac{t}{2}}dt \quad (0 \leq \chi^2 < \infty)$$

$$Q(\chi^2|\nu) = 1 - P(\chi^2|\nu) = \int_{\chi^2}^{\infty} \frac{1}{2^{\frac{\nu}{2}}\Gamma(\frac{\nu}{2})}(t)^{\frac{\nu}{2}-1}e^{-\frac{t}{2}}dt \quad (0 \leq \chi^2 < \infty)$$

(2) **Usage**

Double precision:

```
ierr = ASL_d1cdic  (n, xi, &xo, isw);
```

Single precision:

```
ierr = ASL_r1cdic  (n, xi, &xo, isw);
```

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	n	I	1	Input	Value of number of degrees of freedom ν
2	xi	{ D / R }	1	Input	Value of the cumulative distribution function $P(\chi^2\|\nu)$ or $Q(\chi^2\|\nu)$ of the χ^2 distribution.
3	xo	{ D∗ / R∗ }	1	Output	Value of frequency χ^2
4	isw	I	1	Input	isw=1: Input the value of $P(\chi^2\|\nu)$ for xi isw=2: Input the value of $Q(\chi^2\|\nu)$ for xi
5	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) isw $\in \{1, 2\}$

(b) n ≥ 1

(c) $0.0 \leq$ xi ≤ 1.0

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
1000	xi=0.0 or xi=1.0	0.0 or the positive maximum value is set for xo.
3000	Restriction (a) was not satisfied.	Processing is aborted.
3010	Restriction (b) was not satisfied.	
3020	Restriction (c) was not satisfied.	
3500	The maximum number of iterations was reached before the specified precision was obtained.	The value at that time is returned.
4000	An error occurred in function 3.2.4 $\left\{\begin{array}{l}\text{ASL\_d1cdch}\\ \text{ASL\_r1cdch}\end{array}\right\}$.	Processing is aborted.

(6) **Notes**

(a) The mean and variance of a χ^2 distribution having $n$degrees of freedom are given by the following equations.

$$E[\chi^2(n)] = n, \sigma^2[\chi^2(n)] = 2n$$

(b) If u is the random variable that obeys the standard normal distribution $N(0,1)$, then the distribution of u^2 is the χ^2 distribution having 1 degree of freedom.

(c) If $X_i(i = 1, \cdots, n)$ are the random variables of an arbitrary sample of size n extracted from a normal population $(N(\mu, \sigma^2))$ having mean μ and variance σ^2, the following expressions

$$\frac{\sum_{i=1}^{n}(X_i - \overline{X})^2}{\sigma^2} \text{ and } \frac{n(\overline{X} - \mu)^2}{\sigma^2}$$

obey χ^2 distributions having $n - 1$ and 1 degrees of freedom respectively, and they are mutually independent.

(7) **Example**

(a) Problem
For $\nu = 2$, obtain the values of χ^2 for which $P(\chi^2|\nu) = 0.2$ and $Q(\chi^2|\nu) = 0.2$ occur, respectively.

(b) Input data
xi = 0.2 and n = 2.

(c) Main program

```
/*      C interface example for ASL_d1cdic */

#include <stdio.h>
#include <asl.h>

int main()
{
    int n;
    double xi;
    double xo;
    int isw;
    int ierr;


    n=2;
    xi=0.2;

    printf( "    *** ASL_d1cdic ***\n" );
```

```
        printf( "\n     ** Input **\n\n" );

        printf( "\tn = %6d xi = %8.3g\n", n, xi );

        printf( "\n     ** Output **\n\n" );

        isw=1;
        ierr = ASL_d1cdic(n, xi, &xo, isw);
        printf( "\tierr = %6d\n", ierr );
        printf( "\tValue x Corresponding to P(x,n)=%8.3g\n\n", xo );

        isw=2;
        ierr = ASL_d1cdic(n, xi, &xo, isw);
        printf( "\tierr = %6d\n", ierr );
        printf( "\tValue x Corresponding to Q(x,n)=%8.3g\n\n", xo );

        return 0;
    }
```

(d) Output results

```
        *** ASL_d1cdic ***

         ** Input **

        n =      2 xi =      0.2

         ** Output **

        ierr =      0
        Value x Corresponding to P(x,n)=   0.446

        ierr =      0
        Value x Corresponding to Q(x,n)=    3.22
```

3.2.6 ASL_d1cdnc, ASL_r1cdnc Noncentral χ^2 Distribution

(1) **Function**

For a noncentral χ^2 distribution in which the frequency χ^2 value is x, the number of degrees of freedom is ν, and the noncentrality parameter is λ, the ASL_d1cdnc or ASL_r1cdnc obtains the values of the following functions.

(a) probability density function; p.d.f.

$$f(x|\nu,\lambda) = \begin{cases} \dfrac{e^{-\frac{(x+\lambda)}{2}} x^{\frac{(\nu-2)}{2}}}{2^{\frac{\nu}{2}}} \displaystyle\sum_{k=0}^{\infty} \frac{\lambda^k x^k}{2^{2k} k! \Gamma(\frac{\nu}{2}+k)} & (x>0) \\ 0 & (x \le 0) \end{cases}$$

(b) cumulative distribution function; c.d.f.

$$P(x|\nu,\lambda) = \begin{cases} \displaystyle\sum_{k=0}^{\infty} \frac{e^{-\frac{\lambda}{2}}(\frac{\lambda}{2})^k}{k!} \int_0^x \frac{t^{\frac{(\nu+2k)}{2}-1} e^{-\frac{t}{2}}}{2^{\frac{\nu+2k}{2}} \Gamma(\frac{\nu+2k}{2})} dt & (x>0) \\ 0 & (x \le 0) \end{cases}$$

(c) c.d.f.

$$Q(x|\nu,\lambda) = \begin{cases} 1 - P(x|\nu,\lambda) & (x>0) \\ 1 & (x \le 0) \end{cases}$$

(2) **Usage**

Double precision:

```
ierr = ASL_d1cdnc  (n, xl, xi, &xo, isw);
```

Single precision:

```
ierr = ASL_r1cdnc  (n, xl, xi, &xo, isw);
```

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	n	I	1	Input	Value of number of degrees of freedom ν.
2	xl	{D / R}	1	Input	Value of noncentrality parameter λ.
3	xi	{D / R}	1	Input	Value of frequency χ^2.
4	xo	{D∗ / R∗}	1	Output	Value of the probability density function $f(x\|\nu,\lambda)$ or of the cumulative distribution function $P(x\|\nu,\lambda)$ or $Q(x\|\nu,\lambda)$ of the χ^2 distribution.
5	isw	I	1	Input	Processing switch isw=0:Obtain the value of the probability density function $f(x\|\nu,\lambda)$ for xo isw=1:Obtain the value of the cumulative distribution function $P(x\|\nu,\lambda)$ for xo isw=2:Obtain the value of the cumulative distribution function $Q(x\|\nu,\lambda)$ for xo
6	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) isw $\in \{0, 1, 2\}$

(b) n ≥ 1

(c) xl ≥ 0.0

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
1000	xi ≤ 0.0	0.0 or 1.0 is set for xo.
3000	Restriction (a) was not satisfied.	Processing is aborted.
3010	Restriction (b) or (c) was not satisfied.	
3500	The maximum number of iterations was reached before the specified precision was obtained.	The value at that time is returned.

(6) **Notes**

(a) If the number of degrees of freedom ν or the noncentrality parameter λ is greater than or equal to 1000, the function value may not be obtained.

(b) From the relational expression $P(x|\nu,\lambda) + Q(x|\nu,\lambda) = 1$, it is possible to obtain either $P(x|\nu,\lambda)$ or $Q(x|\nu,\lambda)$ from the other. However, there may be times when cancellation of significant digits occurs, preventing good precision from being obtained.

(c) The mean and variance of a noncentral χ^2 distribution having degrees of freedom ν and noncentrality parameter λ are given by the following equations.

$$E[\chi'^2(\nu,\lambda)] = a \quad , \quad \sigma^2[\chi'^2(\nu,\lambda)] = 2a(1+b)$$

Here, a and b are as follows.

$$a = \nu + \lambda \quad , \quad b = \frac{\lambda}{\nu+\lambda}$$

(d) A noncentral χ^2 distribution having noncentrality parameter $\lambda = 0.0$ matches a χ^2 distribution.

(e) If $X_i (i = 1, \cdots, n)$ are the random variables extracted from a normal population $(N_i(\mu_i, {\sigma_i}^2 = 1))$ having mean μ_i and variance ${\sigma_i}^2 = 1$, respectively, and if Z is defined as follows,

$$Z = \sum_{i=0}^{n} {X_i}^2$$

then Z obeys a noncentral χ^2 distribution having degrees of freedom n and noncentrality parameter λ, where λ is given by the following equation.

$$\lambda = \sum_{i=0}^{n} {\mu_i}^2$$

(7) **Example**

(a) Problem
Let $\nu = 2$, $\lambda = 1.0$, and $x = 5.0$ and obtain the values of the probability density function $f(x|\nu,\lambda)$, and of the cumulative distribution functions $P(x|\nu,\lambda)$ and $Q(x|\nu,\lambda)$.

(b) Input data
n = 2, xl = 1.0 and xi = 5.0.

(c) Main program

```
/*      C interface example for ASL_d1cdnc */

#include <stdio.h>
#include <stdlib.h>
#include <asl.h>

int main()
{
    int n;
    double xl;
    double xi;
    double xo;
    int isw;
    int ierr;


    printf( "    *** ASL_d1cdnc ***\n" );
    printf( "\n     ** Input **\n\n" );

    n = 2;
    xl = 1.0;
    xi = 5.0;
    printf( "\tn = %6d\n", n );
    printf( "\txl = %8.3g\n", xl );
    printf( "\txi = %8.3g\n", xi );
```

```
    isw = 0;
    ierr = ASL_d1cdnc(n, xl, xi, &xo, isw);

    printf( "\n\n     ** Output **\n\n" );
    printf( "\tierr = %6d\n\n", ierr );
    printf( "\tValue of P.D.F = %8.3g\n\n", xo );

    isw = 1;
    ierr = ASL_d1cdnc(n, xl, xi, &xo, isw);

    printf( "\n     ** Output **\n\n" );
    printf( "\tierr = %6d\n\n", ierr );
    printf( "\tValue of C.D.F(1) = %8.3g\n\n", xo );

    isw = 2;
    ierr = ASL_d1cdnc(n, xl, xi, &xo, isw);

    printf( "\n     ** Output **\n\n" );
    printf( "\tierr = %6d\n\n", ierr );
    printf( "\tValue of C.D.F(2) = %8.3g\n\n", xo );

    return 0;
}
```

(d) Output results

```
*** ASL_d1cdnc ***

 ** Input **

n =      2
xl =         1
xi =         5


 ** Output **

ierr =      0

Value of P.D.F =   0.0672


 ** Output **

ierr =      0

Value of C.D.F(1) =    0.811


 ** Output **

ierr =      0

Value of C.D.F(2) =    0.189
```

3.2.7 ASL_d1cdix, ASL_r1cdix
Inverse Noncentral χ^2 Distribution

(1) **Function**

Given the cumulative distribution function (c.d.f.) $P(x|\nu,\lambda)$ or $Q(x|\nu,\lambda)$ of a noncentral χ^2 distribution for which the number of degrees of freedom is ν and the noncentrality parameter is λ, the ASL_d1cdix or ASL_r1cdix obtains the value x of the frequency χ^2 at that time. $P(x|\nu,\lambda)$ and $Q(x|\nu,\lambda)$ are defined by the following equations.

$$P(x|\nu,\lambda)=\sum_{k=0}^{\infty}\frac{e^{-\frac{\lambda}{2}}(\frac{\lambda}{2})^k}{k!}\int_0^x\frac{t^{\frac{(\nu+2k)}{2}-1}e^{-\frac{t}{2}}}{2^{\frac{\nu+2k}{2}}\Gamma(\frac{\nu+2k}{2})}dt\quad(0\le x<\infty)$$

$$Q(x|\nu,\lambda)=1-P(x|\nu,\lambda)\quad(0\le x<\infty)$$

(2) **Usage**

Double precision:

```
ierr = ASL_d1cdix  (n, xl, xi, &xo, isw);
```

Single precision:

```
ierr = ASL_r1cdix  (n, xl, xi, &xo, isw);
```

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	n	I	1	Input	Value of number of degrees of freedom ν.
2	xl	{ D / R }	1	Input	Value of noncentrality parameter λ.
3	xi	{ D / R }	1	Input	Value of the cumulative distribution function $P(x\|\nu,\lambda)$ or $Q(x\|\nu,\lambda)$.
4	xo	{ D∗ / R∗ }	1	Output	Value of frequency χ^2.
5	isw	I	1	Input	Processing switch isw=1:Enter the value of the cumulative distribution function $P(x\|\nu,\lambda)$ for xi isw=2:Enter the value of the cumulative distribution function $Q(x\|\nu,\lambda)$ for xi
6	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) isw $\in \{1, 2\}$

(b) n ≥ 1

(c) $\lambda \geq 0.0$

(d) $0.0 \leq$ xi ≤ 1.0

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
1000	xi = 0.0 or xi = 1.0	0.0 or the maximum value is set for xo.
3000	Restriction (a) was not satisfied.	Processing is aborted.
3010	Restriction (b) or (c) was not satisfied.	
3020	Restriction (d) was not satisfied.	
3500	The upper bound could not be found by the bisection method.	The maximum value is set for xo.
3600	The maximum number of iterations was reached before the specified precision was obtained.	The value at that time is returned.
4000	An error occurred in function 3.2.6 $\left\{\begin{matrix}\text{ASL\_d1cdnc}\\ \text{ASL\_r1cdnc}\end{matrix}\right\}$.	Processing is aborted.

(6) **Notes**

(a) The mean and variance of a noncentral χ^2 distribution having degrees of freedom ν and noncentrality parameter λ are given by the following equations.

$$E[\chi'^2(\nu, \lambda)] = a \quad , \quad \sigma^2[\chi'^2(\nu, \lambda)] = 2a(1 + b)$$

Here, a and b are as follows.

$$a = \nu + \lambda \quad , \quad b = \frac{\lambda}{\nu + \lambda}$$

(b) A noncentral χ^2 distribution having noncentrality parameter $\lambda = 0.0$ matches a χ^2 distribution.

(c) If $X_i (i = 1, \cdots, n)$ are the random variables extracted from a normal population $(N_i(\mu_i, {\sigma_i}^2 = 1))$ having mean μ_i and variance ${\sigma_i}^2 = 1$, respectively, and if Z is defined as follows,

$$Z = \sum_{i=0}^{n} {X_i}^2$$

then Z obeys a noncentral χ^2 distribution having degrees of freedom n and noncentrality parameter λ, where λ is given by the following equation.

$$\lambda = \sum_{i=0}^{n} {\mu_i}^2$$

(7) **Example**

(a) Problem

Let $\nu = 2$ and $\lambda = 1.0$ and obtain the value of x for which the cumulative distribution functions satisfy $P(x|\nu, \lambda) = 0.7$ and $Q(x|\nu, \lambda) = 0.7$.

(b) Input data

n = 2, xl = 1.0 and xi = 0.7.

(c) Main program

```
/*      C interface example for ASL_d1cdix */

#include <stdio.h>
#include <stdlib.h>
#include <asl.h>

int main()
{
    int n;
    double xl;
    double xi;
    double xo;
    int isw;
    int ierr;


    printf( "    *** ASL_d1cdix ***\n" );
    printf( "\n     ** Input **\n\n" );

    n = 2;
    xl = 1.0;
    xi = 0.7;

    printf( "\tn = %6d\n", n );
    printf( "\txl = %8.3g\n", xl );
    printf( "\txi = %8.3g\n", xi );

    isw = 1;
    ierr = ASL_d1cdix(n, xl, xi, &xo, isw);

    printf( "\n\n     ** Output **\n\n" );
    printf( "\tierr = %6d\n\n", ierr );
    printf( "\tValue of x corresponding to P(x;n,xl) = xi : %8.3g\n\n",xo);

    isw = 2;
    ierr = ASL_d1cdix(n, xl, xi, &xo, isw);

    printf( "\n     ** Output **\n\n" );
    printf( "\tierr = %6d\n\n", ierr );
    printf( "\tValue of x corresponding to Q(x;n,xl) = xi : %8.3g\n\n",xo);

    return 0;
}
```

(d) Output results

```
*** ASL_d1cdix ***

 ** Input **

n =      2
xl =        1
xi =      0.7


 ** Output **

ierr =      0

Value of x corresponding to P(x;n,xl) = xi :     3.69


 ** Output **

ierr =      0

Value of x corresponding to Q(x;n,xl) = xi :     1.14
```

3.2.8 ASL_d1cdtb, ASL_r1cdtb
t Distribution

(1) **Function**

For a t distribution having frequency t and number of degrees of freedom ν, the ASL_d1cdtb or ASL_r1cdtb obtains the values of the following functions.

(a) probability density function; p.d.f.

$$f(t|\nu) = \frac{\Gamma(\frac{\nu+1}{2})}{\sqrt{\nu\pi}\Gamma(\frac{\nu}{2})(1+\frac{t^2}{\nu})^{\frac{\nu+1}{2}}}$$

(b) cumulative distribution function; c.d.f.

$$P(t|\nu) = \int_{-\infty}^{t} \frac{\Gamma(\frac{\nu+1}{2})}{\sqrt{\nu\pi}\Gamma(\frac{\nu}{2})(1+\frac{x^2}{\nu})^{\frac{\nu+1}{2}}} dx$$

(c) c.d.f.

$$Q(t|\nu) = \int_{t}^{\infty} \frac{\Gamma(\frac{\nu+1}{2})}{\sqrt{\nu\pi}\Gamma(\frac{\nu}{2})(1+\frac{x^2}{\nu})^{\frac{\nu+1}{2}}} dx$$

(2) **Usage**

Double precision:

```
ierr = ASL_d1cdtb  (n, ti, &to, isw);
```

Single precision:

```
ierr = ASL_r1cdtb  (n, ti, &to, isw);
```

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	n	I	1	Input	Value of number of degrees of freedom ν
2	ti	{D / R}	1	Input	Value of frequency t
3	to	{D∗ / R∗}	1	Output	Value of the probability density function $f(t\|\nu)$ of the t distribution or of the cumulative distribution function $P(t\|\nu)$ or $Q(t\|\nu)$ of the t distribution.
4	isw	I	1	Input	isw=0: Obtain the value of the probability density function $f(t\|\nu)$ for to isw=1: Obtain the value of the cumulative distribution function $P(t\|\nu)$ for to isw=2: Obtain the value of the cumulative distribution function $Q(t\|\nu)$ for to
5	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) n≥1

(b) isw∈{0, 1, 2}

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
3000	Restriction (a) was not satisfied.	Processing is aborted.
3100	Restriction (b) was not satisfied.	
4000	An overflow occurred during the calculation (when (isw=0).	

(6) **Notes**

(a) If u is the random variable that obeys the standard normal distribution $N(0,1)$, χ^2 is the random variable that obeys the χ^2 distribution having ν degrees of freedom, and u and χ^2 are mutually independent, then the distribution of the random variable t given by the following equation obeys t distribution having ν degrees of freedom.

$$t = \frac{u}{\sqrt{\frac{\chi^2}{\nu}}}$$

(7) **Example**

(a) Problem

Let t=5.0 and ν=2 and obtain the values of the probability density function $f(t|\nu)$ and the cumulative distribution functions $P(t|\nu)$ and $Q(t|\nu)$.

(b) Input data

ti=5.0 and n=2.

(c) Main program

```
/*      C interface example for ASL_d1cdtb */

#include <stdio.h>
#include <stdlib.h>
#include <asl.h>

int main()
{
    int n;
    double ti;
    double to;
    int isw;
    int ierr;


    n = 2;
    ti = 5.0;

    printf( "    *** ASL_d1cdtb ***\n" );
    printf( "\n     ** Input **\n\n" );

    printf( "\tn  = %6d\n", n );
    printf( "\tti = %6.3g\n", ti );

    printf( "\n     ** Output **\n\n" );

    isw = 0;
    ierr = ASL_d1cdtb(n, ti, &to, isw);
    printf( "\tierr = %6d\n", ierr );
    printf( "\tValue of P.D.F.    = %8.3g\n\n", to );

    isw = 1;
    ierr = ASL_d1cdtb(n, ti, &to, isw);
    printf( "\tierr = %6d\n", ierr );
    printf( "\tValue of C.D.F.(1) = %8.3g\n\n", to );

    isw = 2;
    ierr = ASL_d1cdtb(n, ti, &to, isw);
    printf( "\tierr = %6d\n", ierr );
    printf( "\tValue of C.D.F.(2) = %8.3g\n", to );

    return 0;
}
```

(d) Output results

```
*** ASL_d1cdtb ***

 ** Input **

n  =      2
ti =      5

 ** Output **

ierr =      0
Value of P.D.F.    =  0.00713

ierr =      0
Value of C.D.F.(1) =    0.981

ierr =      0
Value of C.D.F.(2) =   0.0189
```

3.2.9 ASL_d1cdit, ASL_r1cdit Inverse of t Distribution

(1) **Function**

When given the cumulative distribution function (c.d.f.) $P(t|\nu)$ or $Q(t|\nu)$ of the t distribution for which the number of degrees of freedom is ν, the ASL_d1cdit or ASL_r1cdit obtains the frequency t at that time. $P(t|\nu)$ and $Q(t|\nu)$ are defined by the following equations.

$$P(t|\nu) = \int_{-\infty}^{t} \frac{\Gamma(\frac{\nu+1}{2})}{\sqrt{\nu\pi}\Gamma(\frac{\nu}{2})(1+\frac{x^2}{\nu})^{\frac{\nu+1}{2}}} dx$$

$$Q(t|\nu) = \int_{t}^{\infty} \frac{\Gamma(\frac{\nu+1}{2})}{\sqrt{\nu\pi}\Gamma(\frac{\nu}{2})(1+\frac{x^2}{\nu})^{\frac{\nu+1}{2}}} dx$$

(2) **Usage**

Double precision:

```
ierr = ASL_d1cdit  (n, ti, &to, isw);
```

Single precision:

```
ierr = ASL_r1cdit  (n, ti, &to, isw);
```

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	n	I	1	Input	Value of number of degrees of freedom ν
2	ti	{D / R}	1	Input	Value of the cumulative distribution function $P(t\|\nu)$ or $Q(t\|\nu)$ of the t distribution.
3	to	{D∗ / R∗}	1	Output	Value of frequency t
4	isw	I	1	Input	isw=1: Input the value of $P(t\|\nu)$ for ti isw=2: Input the value of $Q(t\|\nu)$ for ti
5	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) 0.0≤ti≤1.0

(b) n≥1

(c) isw∈{1, 2}

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
1000	ti=0.0 was specified.	Processing continues with the negative minimum value or positive maximum value set for to.
1100	ti=1.0 was specified.	Processing continues with the positive maximum value or negative minimum value set for to.
3000	Restriction (a) was not satisfied.	Processing is aborted.
3100	Restriction (b) was not satisfied.	
3200	Restriction (c) was not satisfied.	

(6) **Notes**

(a) To obtain the percentage point for a two-tailed probability of a t distribution, assign a value that is 1/2 of the two-tailed probability for p.

(b) If u is the random variable that obeys the standard normal distribution $N(0,1)$, χ^2 is the random variable that obeys the χ^2 distribution having ν degrees of freedom, and u and χ^2 are mutually independent, then the distribution of the random variable t given by the following equation obeys t distribution having ν degrees of freedom.

$$t = \frac{u}{\sqrt{\frac{\chi^2}{\nu}}}$$

(7) **Example**

(a) Problem
For ν=2, obtain the values of t for which $P(t|\nu)$=0.2 and $Q(t|\nu)$=0.2 occur, respectively.

(b) Input data
ti=0.2 and n=2.

(c) Main program

```
/*      C interface example for ASL_d1cdit */

#include <stdio.h>
#include <stdlib.h>
#include <asl.h>

int main()
{
    int n;
    double ti;
    double to;
    int isw;
    int ierr;

    n = 2;
    ti = 0.2;

    printf( "    *** ASL_d1cdit ***\n" );
    printf( "\n     ** Input **\n\n" );

    printf( "\tn  = %6d\n", n );
    printf( "\tti = %8.3g\n", ti );

    printf( "\n     ** Output **\n\n" );

    isw = 1;
    ierr = ASL_d1cdit(n, ti, &to, isw);
    printf( "\tierr = %6d\n", ierr );
    printf( "\tValue t corresponding to P(x,n) = %8.3g\n\n", to );
```

```
        isw = 2;
        ierr = ASL_d1cdit(n, ti, &to, isw);
        printf( "\tierr = %6d\n", ierr );
        printf( "\tValue t corresponding to Q(x,n) = %8.3g\n", to );

        return 0;
    }
```

(d) Output results

```
        *** ASL_d1cdit ***

         ** Input **

        n  =      2
        ti =      0.2

         ** Output **

        ierr =      0
        Value t corresponding to P(x,n) =    -1.06

        ierr =      0
        Value t corresponding to Q(x,n) =     1.06
```

3.2.10 ASL_d1cdnt, ASL_r1cdnt Noncentral t Distribution

(1) **Function**

For a noncentral t distribution in which the frequency is t, the number of degrees of freedom is ν, and the noncentrality parameter is δ, the ASL_d1cdnt or ASL_r1cdnt obtains the values of the following functions.

(a) probability density function; p.d.f.

$$f(t|\nu,\delta) = \frac{\nu^{\frac{\nu}{2}} e^{-\frac{\delta^2}{2}}}{\sqrt{\pi}\Gamma(\frac{\nu}{2})(\nu+t^2)^{\frac{(\nu+1)}{2}}} \sum_{k=0}^{\infty} \Gamma(\frac{\nu+k+1}{2}) \frac{\delta^k}{k!} (\frac{2t^2}{\nu+t^2})^{\frac{k}{2}}$$

(b) cumulative distribution function; c.d.f.

$$P(t|\nu,\delta) = \int_{-\infty}^{t} \frac{\nu^{\frac{\nu}{2}} e^{-\frac{\delta^2}{2}}}{\sqrt{\pi}\Gamma(\frac{\nu}{2})(\nu+x^2)^{\frac{(\nu+1)}{2}}} \sum_{k=0}^{\infty} \Gamma(\frac{\nu+k+1}{2}) \frac{\delta^k}{k!} (\frac{2x^2}{\nu+x^2})^{\frac{k}{2}} dx$$

(c) c.d.f.

$$Q(t|\nu,\delta) = 1 - P(t|\nu,\delta)$$

(2) **Usage**

Double precision:

```
ierr = ASL_d1cdnt  (n, del, xi, &xo, isw);
```

Single precision:

```
ierr = ASL_r1cdnt  (n, del, xi, &xo, isw);
```

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	n	I	1	Input	Value of number of degrees of freedom ν.
2	del	{D / R}	1	Input	Value of noncentrality parameter δ
3	xi	{D / R}	1	Input	Value of frequency t
4	xo	{D∗ / R∗}	1	Output	Value of the probability density function $f(t\|\nu,\delta)$ or of the cumulative distribution function $P(t\|\nu,\delta)$ or $Q(t\|\nu,\delta)$ of the t distribution.
5	isw	I	1	Input	Processing switch isw=0:Obtain the value of the probability density function $f(t\|\nu,\delta)$ for xo isw=1:Obtain the value of the cumulative distribution function $P(t\|\nu,\delta)$ for xo isw=2:Obtain the value of the cumulative distribution function $Q(t\|\nu,\delta)$ for xo
6	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) isw $\in \{0, 1, 2\}$

(b) n ≥ 1

(c) del ≥ 0.0

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
1000	xi ≤ 0.0	0.0 or 1.0 is set for xo.
2000	When isw = 0, a solution having sufficient precision could not be found.	The value of probability density function at that time is returned.
3000	Restriction (a) was not satisfied.	Processing is aborted.
3010	Restriction (b) or (c) was not satisfied.	

(6) **Notes**

(a) From the relational expression $P(t|\nu,\delta) + Q(t|\nu,\delta) = 1$, it is possible to obtain either $P(t|\nu,\delta)$ or $Q(t|\nu,\delta)$ from the other. However, there may be times when cancellation of significant digits occurs, preventing good precision from being obtained.

(b) A noncentral t distribution having noncentrality parameter $\delta = 0.0$ matches a t distribution.

(7) **Example**

(a) Problem
Let $\nu = 2$, $\delta = 1.0$ and $x = 5.0$ and obtain the values of the probability density function $f(t|\nu,\delta)$ and of the cumulative distribution functions $P(t|\nu,\delta)$ and $Q(t|\nu,\delta)$.

(b) Input data
n = 2, del = 1.0 and xi = 5.0.

(c) Main program

```
/*      C interface example for ASL_d1cdnt */

#include <stdio.h>
#include <stdlib.h>
#include <asl.h>

int main()
{
    int n;
    double del;
    double xi;
    double xo;
    int isw;
    int ierr;


    printf( "    *** ASL_d1cdnt ***\n" );
    printf( "\n     ** Input **\n\n" );

    n = 2;
    del = 1.0;
    xi = 5.0;

    printf( "\tn = %6d\n", n );
    printf( "\tdel = %8.3g\n", del );
    printf( "\txi  = %8.3g\n", xi );

    isw = 0;
    ierr = ASL_d1cdnt(n, del, xi, &xo, isw);

    printf( "\n\n     ** Output **\n\n" );
    printf( "\tierr = %6d\n\n", ierr );
    printf( "\tValue of P.D.F = %8.3g\n\n", xo );

    isw = 1;
    ierr = ASL_d1cdnt(n, del, xi, &xo, isw);

    printf( "\n     ** Output **\n\n" );
    printf( "\tierr = %6d\n\n", ierr );
    printf( "\tValue of C.D.F(1) = %8.3g\n\n", xo );

    isw = 2;
    ierr = ASL_d1cdnt(n, del, xi, &xo, isw);

    printf( "\n     ** Output **\n\n" );
    printf( "\tierr = %6d\n\n", ierr );
    printf( "\tValue of C.D.F(2) = %8.3g\n\n", xo );

    return 0;
}
```

(d) Output results

```
*** ASL_d1cdnt ***

 ** Input **

n =      2
del =        1
xi  =        5
```

```
 ** Output **
ierr =      0
Value of P.D.F =   0.0253

 ** Output **
ierr =      0
Value of C.D.F(1) =     0.93

 ** Output **
ierr =      0
Value of C.D.F(2) =   0.0698
```

3.2.11 ASL_d1cdis, ASL_r1cdis
Inverse Noncentral t Distribution

(1) **Function**

Given the cumulative distribution function (c.d.f.) $P(t|\nu,\delta)$ or $Q(t|\nu,\delta)$ of a noncentral t distribution for which the number of degrees of freedom is ν and the noncentrality parameter is δ, the ASL_d1cdis or ASL_r1cdis obtains the value of the frequency t at that time. $P(t|\nu,\delta)$ and $Q(t|\nu,\delta)$ are defined by the following equations.

$$P(t|\nu,\delta) = \int_{-\infty}^{t} \frac{\nu^{\frac{\nu}{2}} e^{-\frac{\delta^2}{2}}}{\sqrt{\pi}\Gamma(\frac{\nu}{2})(\nu+x^2)^{\frac{(\nu+1)}{2}}} \sum_{k=0}^{\infty} \Gamma(\frac{\nu+k+1}{2}) \frac{\delta^k}{k!} (\frac{2x^2}{\nu+x^2})^{\frac{k}{2}} dx$$

$$Q(t|\nu,\delta) = 1 - P(t|\nu,\delta)$$

(2) **Usage**

Double precision:

```
ierr = ASL_d1cdis  (n, del, xi, &xo, isw);
```

Single precision:

```
ierr = ASL_r1cdis  (n, del, xi, &xo, isw);
```

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	n	I	1	Input	Value of number of degrees of freedom ν.
2	del	{ D / R }	1	Input	Value of noncentrality parameter δ.
3	xi	{ D / R }	1	Input	Value of the cumulative distribution function $P(t\|\nu,\delta)$ or $Q(t\|\nu,\delta)$ of the t distribution.
4	xo	{ D∗ / R∗ }	1	Output	Value of frequency t.
5	isw	I	1	Input	Processing switch isw=1:Input the value of the cumulative distribution function $P(t\|\nu,\delta)$ for xi isw=2:Input the value of the cumulative distribution function $Q(t\|\nu,\delta)$ for xi
6	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) isw $\in \{0, 1, 2\}$

(b) n ≥ 1

(c) $\delta \geq 0.0$

(d) $0.0 \leq$ xi ≤ 1.0

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
1000	xi = 0.0 or xi = 1.0	The minimum value or the maximum value is set for xo.
3000	Restriction (a) was not satisfied.	Processing is aborted.
3010	Restriction (b) or (c) was not satisfied.	
3020	Restriction (d) was not satisfied.	
3510	The lower bound could not be found by the bisection method.	The minimum value is set for xo.
3520	The upper bound could not be found by the bisection method.	The maximum value is set for xo.
3600	The maximum number of iterations was reached before the specified precision was obtained.	The value at that time is returned.
4000	An error occurred in function 3.2.10 {ASL_d1cdnt, ASL_r1cdnt}.	Processing is aborted.

(6) **Notes**

(a) A noncentral t distribution having noncentrality parameter $\delta = 0.0$ matches a t distribution.

(7) **Example**

(a) Problem
Let $\nu = 2$ and $\delta = 1.0$ and obtain the value of t for which the cumulative distribution functions satisfy $P(t|\nu, \delta) = 0.7$ and $Q(t|\nu, \delta) = 0.7$.

(b) Input data
n = 2, del = 1.0 and xi = 0.7.

(c) Main program

```
/*      C interface example for ASL_d1cdis */

#include <stdio.h>
#include <stdlib.h>
#include <asl.h>

int main()
{
    int n;
    double del;
    double xi;
    double xo;
    int isw;
    int ierr;
```

```
    printf( "    *** ASL_d1cdis ***\n" );
    printf( "\n     ** Input **\n\n" );

    n = 2;
    del = 1.0;
    xi = 0.7;

    printf( "\tn   = %6d\n", n );
    printf( "\tdel = %8.3g\n", del );
    printf( "\txi  = %8.3g\n", xi );

    isw = 1;
    ierr = ASL_d1cdis(n, del, xi, &xo, isw);

    printf( "\n\n     ** Output **\n\n" );
    printf( "\tierr = %6d\n\n", ierr );
    printf( "\tValue of x corresponding to P(x;n,del)=xi : %8.3g\n\n", xo );

    isw = 2;
    ierr = ASL_d1cdis(n, del, xi, &xo, isw);

    printf( "\n     ** Output **\n\n" );
    printf( "\tierr = %6d\n\n", ierr );
    printf( "\tValue of x corresponding to Q(x;n,del)=xi : %8.3g\n\n", xo );

    return 0;
}
```

(d) Output results

```
*** ASL_d1cdis ***

 ** Input **

n   =      2
del =        1
xi  =      0.7


 ** Output **

ierr =      0

Value of x corresponding to P(x;n,del)=xi :     1.96


 ** Output **

ierr =      0

Value of x corresponding to Q(x;n,del)=xi :    0.521
```

3.2.12 ASL_d1cdfb, ASL_r1cdfb
F Distribution

(1) **Function**

For a F distribution having frequency F and numbers of degrees of freedom ν_1 and ν_2, the ASL_d1cdfb or ASL_r1cdfb obtains the values of the following functions.

(a) probability density function; p.d.f.

$$f(F|\nu_1,\nu_2) = \frac{\nu_1^{\frac{\nu_1}{2}} \cdot \nu_2^{\frac{\nu_2}{2}} \cdot x^{\frac{\nu_1}{2}-1}}{B(\frac{\nu_1}{2},\frac{\nu_2}{2})(\nu_1 x+\nu_2)^{\frac{\nu_1+\nu_2}{2}}}$$

(b) cumulative distribution function; c.d.f.

$$P(F|\nu_1,\nu_2) = \int_0^F \frac{\nu_1^{\frac{\nu_1}{2}} \cdot \nu_2^{\frac{\nu_2}{2}} \cdot x^{\frac{\nu_1}{2}-1}}{B(\frac{\nu_1}{2},\frac{\nu_2}{2})(\nu_1 x+\nu_2)^{\frac{\nu_1+\nu_2}{2}}} dx$$

(c) c.d.f.

$$Q(F|\nu_1,\nu_2) = \int_F^\infty \frac{\nu_1^{\frac{\nu_1}{2}} \cdot \nu_2^{\frac{\nu_2}{2}} \cdot x^{\frac{\nu_1}{2}-1}}{B(\frac{\nu_1}{2},\frac{\nu_2}{2})(\nu_1 x+\nu_2)^{\frac{\nu_1+\nu_2}{2}}} dx$$

(2) **Usage**

Double precision:

```
ierr = ASL_d1cdfb  (n1, n2, fi, &fo, isw);
```

Single precision:

```
ierr = ASL_r1cdfb  (n1, n2, fi, &fo, isw);
```

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	n1	I	1	Input	Number of degrees of freedom ν_1
2	n2	I	1	Input	Number of degrees of freedom ν_2
3	fi	{D / R}	1	Input	Value of frequency F
4	fo	{D∗ / R∗}	1	Output	Value of the probability density function $f(F\|\nu_1,\nu_2)$ of the F distribution or of the cumulative distribution function $P(F\|\nu_1,\nu_2)$ or $Q(F\|\nu_1,\nu_2)$ of the F distribution.
5	isw	I	1	Input	isw=0: Obtain the value of the probability density function $f(F\|\nu_1,\nu_2)$ for fo isw=1: Obtain the value of the cumulative distribution function $P(F\|\nu_1,\nu_2)$ for fo isw=2: Obtain the value of the cumulative distribution function $Q(F\|\nu_1,\nu_2)$ for fo
6	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) f>0.0

(b) n1≥1, n2≥1

(c) isw∈{0, 1, 2}

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
2000	n1>2000, n2>2000 (when isw=0)	Processing terminated, but the precision of the solution is low.
3000	Restriction (a) was not satisfied.	Processing is aborted.
3100	Restriction (b) was not satisfied.	
3200	Restriction (c) was not satisfied.	
4000	$\|B(\frac{n_1}{2}, \frac{n_2}{2})\|$ <(Positive minimum value) (when isw=0)	
4100	An error occurred in the step for obtaining the beta function value. (when isw=0)	
4200	An overflow occurred during the calculation. (when isw=0)	

(6) **Notes**

(a) If χ_1^2 and χ_2^2 are the random variables that obey the χ^2 distribution having ν_1 and ν_2 degrees of freedom respectively and χ_1^2 and χ_2^2 are mutually independent, then the distribution of the random variable F given by the following equation obeys F distribution having ν_1 and ν_2 degrees of freedom.

$$F = \frac{\frac{\chi_1^2}{\nu_1}}{\frac{\chi_2^2}{\nu_2}}$$

(7) **Example**

(a) Problem

Let F=5.0, ν_1=2 and ν_2=2 and obtain the values of the probability density function $f(F|\nu_1, \nu_2)$ and the cumulative distribution functions $P(F|\nu_1, \nu_2)$ and $Q(F|\nu_1, \nu_2)$.

(b) Input data

fi=5.0, n1=2 and n2=2.

(c) Main program

```
/*      C interface example for ASL_d1cdfb */

#include <stdio.h>
#include <stdlib.h>
#include <asl.h>

int main()
{
    int n1;
    int n2;
    double fi;
    double fo;
    int isw;
    int ierr;


    n1 = 2;
    n2 = 2;
    fi = 5.0;

    printf( "    *** ASL_d1cdfb ***\n" );
    printf( "\n     ** Input **\n\n" );

    printf( "\tn1 = %6d\n", n1 );
    printf( "\tn2 = %6d\n", n2 );
    printf( "\tfi = %6.3g\n", fi );

    printf( "\n     ** Output **\n\n" );

    isw=0;
    ierr = ASL_d1cdfb(n1, n2, fi, &fo, isw);
    printf( "\tierr = %6d\n", ierr );
    printf( "\tValue of P.D.F.    = %8.3g\n\n", fo );

    isw=1;
    ierr = ASL_d1cdfb(n1, n2, fi, &fo, isw);
    printf( "\tierr = %6d\n", ierr );
    printf( "\tValue of C.D.F.(1) = %8.3g\n\n", fo );

    isw=2;
    ierr = ASL_d1cdfb(n1, n2, fi, &fo, isw);
    printf( "\tierr = %6d\n", ierr );
    printf( "\tValue of C.D.F.(2) = %8.3g\n", fo );

    return 0;
}
```

(d) Output results

```
*** ASL_d1cdfb ***

 ** Input **

n1 =      2
n2 =      2
fi =      5

 ** Output **

ierr =      0
```

```
Value of P.D.F.    =   0.0278

ierr =       0
Value of C.D.F.(1) =    0.833

ierr =       0
Value of C.D.F.(2) =    0.167
```

3.2.13 ASL_d1cdif, ASL_r1cdif Inverse of F Distribution

(1) **Function**

When given the cumulative distribution function (c.d.f.) $P(F|\nu_1,\nu_2)$ or $Q(F|\nu_1,\nu_2)$ of the F distribution for which the numbers of degrees of freedom are ν_1 and ν_2, the ASL_d1cdif or ASL_r1cdif obtains the frequency F at that time. $P(F|\nu_1,\nu_2)$ and $Q(F|\nu_1,\nu_2)$ are defined by the following equations.

$$P(F|\nu_1,\nu_2)=\int_0^F \frac{\nu_1^{\frac{\nu_1}{2}}\cdot\nu_2^{\frac{\nu_2}{2}}\cdot x^{\frac{\nu_1}{2}-1}}{B(\frac{\nu_1}{2},\frac{\nu_2}{2})(\nu_1 x+\nu_2)^{\frac{\nu_1+\nu_2}{2}}}dx$$

$$Q(F|\nu_1,\nu_2)=\int_F^\infty \frac{\nu_1^{\frac{\nu_1}{2}}\cdot\nu_2^{\frac{\nu_2}{2}}\cdot x^{\frac{\nu_1}{2}-1}}{B(\frac{\nu_1}{2},\frac{\nu_2}{2})(\nu_1 x+\nu_2)^{\frac{\nu_1+\nu_2}{2}}}dx$$

(2) **Usage**

Double precision:

```
ierr = ASL_d1cdif  (n1, n2, fi, &fo, isw);
```

Single precision:

```
ierr = ASL_r1cdif  (n1, n2, fi, &fo, isw);
```

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	n1	I	1	Input	Number of degrees of freedom ν_1
2	n2	I	1	Input	Number of degrees of freedom ν_2
3	fi	{D / R}	1	Input	Value of the cumulative distribution function $P(F\|\nu_1,\nu_2)$ or $Q(F\|\nu_1,\nu_2)$ of the F distribution.
4	fo	{D∗ / R∗}	1	Output	Value of frequency F
5	isw	I	1	Input	isw=1: Assign the value of $P(F\|\nu_1,\nu_2)$ for fi isw=2: Assign the value of $Q(F\|\nu_1,\nu_2)$ for fi
6	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) 0.0≤fi≤1.0

(b) n1≥1, n2≥1

(c) isw∈{1, 2}

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
1000	fi=0.0 was specified.	Processing continues with 0.0 or the positive maximum value set for fo.
1100	fi=1.0 was specified.	Processing continues with the positive maximum value or 0.0 set for fo.
3000	Restriction (a) was not satisfied.	Processing is aborted.
3100	Restriction (b) was not satisfied.	
3200	Restriction (c) was not satisfied.	

(6) **Notes**

(a) If χ_1^2 and χ_2^2 are the random variables that obey the χ^2 distribution having ν_1 and ν_2 degrees of freedom respectively and χ_1^2 and χ_2^2 are mutually independent, then the distribution of the random variable F given by the following equation obeys F distribution having ν_1 and ν_2 degrees of freedom.

$$F = \frac{\frac{\chi_1^2}{\nu_1}}{\frac{\chi_2^2}{\nu_2}}$$

(7) **Example**

(a) Problem
For ν_1=2 and ν_2=2 obtain the values of F for which $P(F|\nu_1,\nu_2)$=0.2 and $Q(F|\nu_1,\nu_2)$=0.2 occur, respectively.

(b) Input data
fi=0.2, n1=2 and n2=2.

(c) Main program

```
/*      C interface example for ASL_d1cdif */

#include <stdio.h>
#include <stdlib.h>
#include <asl.h>

int main()
{
    int n1;
    int n2;
    double fi;
    double fo;
    int isw;
    int ierr;


    n1 = 2;
    n2 = 2;
    fi = 0.2;

    printf( "    *** ASL_d1cdif ***\n" );
    printf( "\n     ** Input **\n\n" );

    printf( "\tn1  = %6d\n", n1 );
    printf( "\tn2  = %6d\n", n2 );
    printf( "\tfi  = %8.3g\n", fi );

    printf( "\n     ** Output **\n\n" );

    isw = 1;
    ierr = ASL_d1cdif(n1, n2, fi, &fo, isw);
    printf( "\tierr = %6d\n", ierr );
    printf( "\tValue t corresponding to P(x,n) = %8.3g\n\n", fo );

    isw = 2;
    ierr = ASL_d1cdif(n1, n2, fi, &fo, isw);
    printf( "\tierr = %6d\n", ierr );
```

```
        printf( "\tValue t corresponding to Q(x,n) = %8.3g\n", fo );

        return 0;
    }
```

(d) Output results

```
        *** ASL_d1cdif ***

         ** Input **

        n1  =      2
        n2  =      2
        fi  =      0.2

         ** Output **

        ierr =      0
        Value t corresponding to P(x,n) =     0.25

        ierr =      0
        Value t corresponding to Q(x,n) =        4
```

3.2.14 ASL_d1cdgm, ASL_r1cdgm Gamma Distribution

(1) **Function**

For a gamma distribution having parameters α and β, the ASL_d1cdgm or ASL_r1cdgm obtains the values of the following functions.

(a) probability density function; p.d.f.

$$f(x;\alpha,\beta) = \begin{cases} \dfrac{\beta^{\alpha}}{\Gamma(\alpha)} x^{\alpha-1} e^{-\beta x} & (x > 0; \alpha, \beta > 0) \\ 0 & (x \leq 0; \alpha, \beta > 0) \end{cases}$$

(b) cumulative distribution function; c.d.f.

$$P(x;\alpha,\beta) = \int_0^x \frac{\beta^{\alpha}}{\Gamma(\alpha)} t^{\alpha-1} e^{-\beta t} dt \quad (\alpha, \beta > 0)$$

(c) c.d.f.

$$Q(x;\alpha,\beta) = 1 - P(x;\alpha,\beta) = \int_x^{\infty} \frac{\beta^{\alpha}}{\Gamma(\alpha)} t^{\alpha-1} e^{-\beta t} dt \quad (\alpha, \beta > 0)$$

(2) **Usage**

Double precision:

```
ierr = ASL_d1cdgm  (a, b, xi, &xo, isw);
```

Single precision:

```
ierr = ASL_r1cdgm  (a, b, xi, &xo, isw);
```

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	a	{D / R}	1	Input	Value of shape parameter α
2	b	{D / R}	1	Input	Value of scale parameter β
3	xi	{D / R}	1	Input	Value of random variable x
4	xo	{D∗ / R∗}	1	Output	Value of the probability density function $f(x;\alpha,\beta)$ of the gamma distribution or of the cumulative distribution function $P(x;\alpha,\beta)$ or $Q(x;\alpha,\beta)$ of the gamma distribution.
5	isw	I	1	Input	isw=0: Obtain the value of the probability density function $f(x;\alpha,\beta)$ for xo isw=1: Obtain the value of the cumulative distribution function $P(x;\alpha,\beta)$ for xo isw=2: Obtain the value of the cumulative distribution function $Q(x;\alpha,\beta)$ for xo
6	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) isw $\in \{0, 1, 2\}$

(b) a > 0.0

(c) b > 0.0

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
1000	xi ≤ 0.0	0.0 or 1.0 is set for xo.
3000	Restriction (a) was not satisfied.	Processing is aborted.
3010	Restriction (b) was not satisfied.	
3020	Restriction (c) was not satisfied.	

(6) **Notes**

(a) From the relational expression $P(x;\alpha,\beta)+Q(x;\alpha,\beta)=1$, it is possible to obtain either $P(x;\alpha,\beta)$ or $Q(x;\alpha,\beta)$ from the other. However, there may be times when cancellation of significant digits occurs, preventing good precision from being obtained.

(b) The mean and variance of a gamma distribution having parameters α and β are given by the following equations.

$$E[x]=\frac{\alpha}{\beta},\sigma^2[x]=\frac{\alpha}{\beta^2}$$

(c) A gamma distribution for which $\alpha=1$ is an exponential distribution. Also, a distribution for which the value of α is limited to a positive integer is called an Erlang distribution.

(7) **Example**

(a) Problem
Let $\alpha=5.0$, $\beta=2.0$ and $x=3.0$ and obtain the values of the probability density function $f(x;\alpha,\beta)$ and the cumulative distribution functions $P(x;\alpha,\beta)$ and $Q(x;\alpha,\beta)$.

(b) Input data
a = 5.0, b = 2.0 and xi = 3.0.

(c) Main program

```
/*      C interface example for ASL_d1cdgm */

#include <stdio.h>
#include <asl.h>

int main()
{
    double a;
    double b;
    double xi;
    double xo;
    int isw;
    int ierr;


    a=5.0;
    b=2.0;
    xi=3.0;

    printf( "    *** ASL_d1cdgm ***\n" );
    printf( "\n     ** Input **\n\n" );
    printf( "\ta = %8.3g b = %8.3g xi = %8.3g\n", a, b, xi );
    printf( "\n     ** Output **\n\n" );

    isw=0;
    ierr = ASL_d1cdgm(a, b, xi, &xo, isw);

    printf( "\tierr = %6d\n", ierr );
    printf( "\tValue of P.D.F   =%8.3g\n\n", xo );
    isw=1;
    ierr = ASL_d1cdgm(a, b, xi, &xo, isw);

    printf( "\tierr = %6d\n", ierr );
    printf( "\tValue of C.D.F(1)=%8.3g\n\n", xo );
    isw=2;
    ierr = ASL_d1cdgm(a, b, xi, &xo, isw);

    printf( "\tierr = %6d\n", ierr );
    printf( "\tValue of C.D.F(2)=%8.3g\n\n", xo );

    return 0;
}
```

(d) Output results

```
*** ASL_d1cdgm ***

 ** Input **

a =        5 b =        2 xi =        3

 ** Output **
```

```
ierr =      0
Value of P.D.F   =   0.268

ierr =      0
Value of C.D.F(1)=   0.715

ierr =      0
Value of C.D.F(2)=   0.285
```

3.2.15 ASL_d1cdig, ASL_r1cdig
Inverse Gamma Distribution

(1) **Function**

Given the (cumulative distribution function; c.d.f.) $P(x;\alpha,\beta)$ or $Q(x;\alpha,\beta)$ of a gamma distribution having parameters α and β, the ASL_d1cdig or ASL_r1cdig obtains the value x of the random variable at that time. $P(x;\alpha,\beta)$ and $Q(x;\alpha,\beta)$ are defined by the following equations.

$$P(x;\alpha,\beta) = \int_0^x \frac{\beta^\alpha}{\Gamma(\alpha)} t^{\alpha-1} e^{-\beta t} dt \quad (\alpha,\beta > 0)$$

$$Q(x;\alpha,\beta) = 1 - P(x;\alpha,\beta) = \int_x^\infty \frac{\beta^\alpha}{\Gamma(\alpha)} t^{\alpha-1} e^{-\beta t} dt \quad (\alpha,\beta > 0)$$

(2) **Usage**

Double precision:

```
ierr = ASL_d1cdig  (a, b, xi, &xo, isw);
```

Single precision:

```
ierr = ASL_r1cdig  (a, b, xi, &xo, isw);
```

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	a	{D / R}	1	Input	Value of shape parameter α.
2	b	{D / R}	1	Input	Value of scale parameter β.
3	xi	{D / R}	1	Input	Value of the cumulative distribution function $P(x;\alpha,\beta)$ or $Q(x;\alpha,\beta)$ of the gamma distribution.
4	xo	{D∗ / R∗}	1	Output	Value of random variable x.
5	isw	I	1	Input	Processing switch isw=1:Input the value of the cumulative distribution function $P(x;\alpha,\beta)$ for xi isw=2:Input the value of the cumulative distribution function $Q(x;\alpha,\beta)$ for xi
6	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) isw $\in \{1, 2\}$

(b) a, b > 0.0

(c) $0.0 \leq$ xi ≤ 1.0

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
1000	xi = 0.0 or xi = 1.0	If xi = 0.0 When isw=1: 0.0 is set for xo. When isw=2:The maximum value is set for xo. If xi = 1.0 When isw=1:The maximum value is set for xo. When isw=2:0.0 is set for xo.
3000	Restriction (a) was not satisfied.	Processing is aborted.
3010	Restriction (b) was not satisfied.	
3020	Restriction (c) was not satisfied.	
3500	The upper bound could not be found by the bisection method.	The maximum value is set for xo.
3600	The maximum number of iterations was reached before the specified precision was obtained.	The value at that time is returned.
4000	An error occurred in function 3.2.14 {ASL_d1cdgm / ASL_r1cdgm}.	Processing is aborted.

(6) **Notes**

(a) The mean and variance of a gamma distribution having parameters α and β are given by the following equations.

$$E[x] = \frac{\alpha}{\beta}, \sigma^2[x] = \frac{\alpha}{\beta^2}$$

(b) A gamma distribution with $\alpha = 1$ is an exponential distribution. A distribution for which the value of α is limited to positive integers is called an Erlang distribution.

(7) **Example**

(a) Problem
For $\alpha = 5.0$ and $\beta = 2.0$, obtain the values of x for which the cumulative distribution function $P(x; \alpha, \beta) = 0.7$, or $Q(x; \alpha, \beta) = 0.7$, respectively.

(b) Input data
a = 5.0, b = 2.0 and xi = 0.7.

(c) Main program

```
/*      C interface example for ASL_d1cdig */

#include <stdio.h>
#include <stdlib.h>
#include <asl.h>

int main()
{
    double a;
    double b;
    double xi;
    double xo;
    int isw;
    int ierr;


    printf( "    *** ASL_d1cdig ***\n" );
    printf( "\n     ** Input **\n\n" );

    a = 5.0;
    b = 2.0;
    xi = 0.7;

    printf( "\ta  = %8.3g\n", a );
    printf( "\tb  = %8.3g\n", b );
    printf( "\txi = %8.3g\n", xi );

    isw = 1;
    ierr = ASL_d1cdig(a, b, xi, &xo, isw);

    printf( "\n\n     ** Output **\n\n" );
    printf( "\tierr = %6d\n\n", ierr );
    printf( "\tValue of x corresponding to P(x;a,b) = xi  %8.3g\n\n", xo );

    isw = 2;
    ierr = ASL_d1cdig(a, b, xi, &xo, isw);

    printf( "\n     ** Output **\n\n" );
    printf( "\tierr = %6d\n\n", ierr );
    printf( "\tValue of x corresponding to Q(x;a,b) = xi  %8.3g\n", xo );

    return 0;
}
```

(d) Output results

```
*** ASL_d1cdig ***

 ** Input **

a  =        5
b  =        2
xi =      0.7


 ** Output **

ierr =      0

Value of x corresponding to P(x;a,b) = xi      2.95


 ** Output **

ierr =      0

Value of x corresponding to Q(x;a,b) = xi      1.82
```

3.2.16 ASL_d1cdbt, ASL_r1cdbt Beta Distribution

(1) **Function**

For a beta distribution having the two positive numbers a and b as parameters, the ASL_d1cdbt or ASL_r1cdbt obtains the values of the following functions.

(a) probability density function; p.d.f.

$$f(x;a,b) = \begin{cases} \dfrac{1}{B(a,b)} x^{a-1}(1-x)^{b-1} & (0 < x < 1; a, b > 0) \\ 0 & (x \leq 0, x \geq 1; a, b > 0) \end{cases}$$

(b) cumulative distribution function; c.d.f.

$$P(x;a,b) = \begin{cases} 0 & (x \leq 0; a, b > 0) \\ \dfrac{1}{B(a,b)} \displaystyle\int_0^x t^{a-1}(1-t)^{b-1} dt & (0 < x < 1; a, b > 0) \\ 1 & (x \geq 1; a, b > 0) \end{cases}$$

(c) c.d.f.

$$\begin{aligned} Q(x;a,b) &= 1 - P(x;a,b) \\ &= \begin{cases} 1 & (x \leq 0; a, b > 0) \\ \dfrac{1}{B(a,b)} \displaystyle\int_x^1 t^{a-1}(1-t)^{b-1} dt & (0 < x < 1; a, b > 0) \\ 0 & (x \geq 1; a, b > 0) \end{cases} \end{aligned}$$

(2) **Usage**

Double precision:

```
ierr = ASL_d1cdbt  (a, b, xi, &xo, isw);
```

Single precision:

```
ierr = ASL_r1cdbt  (a, b, xi, &xo, isw);
```

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	a	{D / R}	1	Input	Value of shape parameter a.
2	b	{D / R}	1	Input	Value of shape parameter b.
3	xi	{D / R}	1	Input	Value of random variable x
4	xo	{D∗ / R∗}	1	Output	Value of the probability density function $f(x;a,b)$ or of the cumulative distribution function $P(x;a,b)$ or $Q(x;a,b)$ of the beta distribution
5	isw	I	1	Input	Processing switch isw=0:Obtain the value of the probability density function $f(x;a,b)$ for xo isw=1:Obtain the value of the cumulative distribution function $P(x;a,b)$ for xo isw=2:Obtain the value of the cumulative distribution function $Q(x;a,b)$ for xo
6	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) isw $\in \{0, 1, 2\}$

(b) a, b > 0.0

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
1000	xi ≤ 0.0 or xi ≥ 1.0	0.0 or 1.0 is set for xo.
3000	Restriction (a) was not satisfied.	Processing is aborted.
3010	Restriction (b) was not satisfied.	
3500	The maximum number of iterations was reached before the specified precision was obtained.	The value at that time is returned.

(6) **Notes**

(a) From the relational expression $P(x;a,b) + Q(x;a,b) = 1$, it is possible to obtain either $P(x;a,b)$ or $Q(x;a,b)$ from the other. However, there may be times when cancellation of significant digits occurs, preventing good precision from being obtained.

(b) The mean and variance of a beta distribution having parameters a and b are given by the following equations.

$$E[x] = \frac{a}{a+b}, \quad \sigma^2[x] = \frac{ab}{(a+b)^2(a+b+1)}$$

(c) A beta distribution with $a = b = 1$ is a uniform distribution on the interval (0, 1).

(7) **Example**

(a) Problem
Let $a = 5.0$, $b = 2.0$ and $x = 0.3$ and obtain the values of the probability density function $f(x;a,b)$ and the cumulative distribution functions $P(x;a,b)$ and $Q(x;a,b)$.

(b) Input data
a = 5.0, b = 2.0 and xi = 0.3.

(c) Main program

```
/*      C interface example for ASL_d1cdbt */

#include <stdio.h>
#include <stdlib.h>
#include <asl.h>

int main()
{
    double a;
    double b;
    double xi;
    double xo;
    int isw;
    int ierr;


    printf( "    *** ASL_d1cdbt ***\n" );
    printf( "\n     ** Input **\n\n" );

    a = 5.0;
    b = 2.0;
    xi = 0.3;

    printf( "\ta  = %8.3g\n", a );
    printf( "\tb  = %8.3g\n", b );
    printf( "\txi = %8.3g\n", xi );

    isw = 0;
    ierr = ASL_d1cdbt(a, b, xi, &xo, isw);

    printf( "\n\n     ** Output **\n\n" );
    printf( "\tierr = %6d\n\n", ierr );
    printf( "\tValue of P.D.F = %8.3g\n\n", xo );

    isw = 1;
    ierr = ASL_d1cdbt(a, b, xi, &xo, isw);

    printf( "\n     ** Output **\n\n" );
    printf( "\tierr = %6d\n\n", ierr );
    printf( "\tValue of C.D.F(1) = %8.3g\n\n", xo );

    isw = 2;
    ierr = ASL_d1cdbt(a, b, xi, &xo, isw);

    printf( "\n     ** Output **\n\n" );
    printf( "\tierr = %6d\n\n", ierr );
    printf( "\tValue of C.D.F(2) = %8.3g\n\n", xo );

    return 0;
}
```

(d) Output results

```
*** ASL_d1cdbt ***

 ** Input **

a  =         5
b  =         2
xi =       0.3


 ** Output **

ierr =       0

Value of P.D.F =      0.17


 ** Output **

ierr =       0

Value of C.D.F(1) =    0.0109


 ** Output **

ierr =       0

Value of C.D.F(2) =     0.989
```

3.2.17 ASL_d1cdib, ASL_r1cdib Inverse Beta Distribution

(1) **Function**

Given the (cumulative distribution function; c.d.f.) $P(x;\alpha,\beta)$ or $Q(x;\alpha,\beta)$ of a beta distribution having positive parameters a and b, the ASL_d1cdib or ASL_r1cdib obtains the value x of the random variable at that time. $P(x;\alpha,\beta)$ and $Q(x;\alpha,\beta)$ are defined by the following equations.

$$P(x;a,b) = \begin{cases} 0 & (x \le 0; a, b > 0) \\ \dfrac{1}{B(a,b)} \displaystyle\int_0^x t^{a-1}(1-t)^{b-1}dt & (0 < x < 1; a, b > 0) \\ 1 & (x \ge 1; a, b > 0) \end{cases}$$

$$Q(x;a,b) = 1 - P(x;a,b)$$

$$= \begin{cases} 1 & (x \le 0; a, b > 0) \\ \dfrac{1}{B(a,b)} \displaystyle\int_x^\infty t^{a-1}(1-t)^{b-1}dt & (0 < x < 1; a, b > 0) \\ 0 & (x \ge 1; a, b > 0) \end{cases}$$

(2) **Usage**

Double precision:

```
ierr = ASL_d1cdib  (a, b, xi, &xo, isw);
```

Single precision:

```
ierr = ASL_r1cdib  (a, b, xi, &xo, isw);
```

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	a	{D / R}	1	Input	Value of shape parameter a.
2	b	{D / R}	1	Input	Value of shape parameter b.
3	xi	{D / R}	1	Input	Value of the cumulative distribution function $P(x;a,b)$ or $Q(x;a,b)$ of the beta distribution.
4	xo	{D∗ / R∗}	1	Output	Value of random variable x.
5	isw	I	1	Input	Processing switch isw=1:Input the value of the cumulative distribution function $P(x;a,b)$ for xi isw=2:Input the value of the cumulative distribution function $Q(x;a,b)$ for xi
6	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) isw $\in \{1, 2\}$

(b) a, b > 0.0

(c) $0.0 \leq$ xi ≤ 1.0

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
1000	xi = 0.0 or xi = 1.0	0.0 or 1.0 is set for xo.
3000	Restriction (a) was not satisfied.	Processing is aborted.
3010	Restriction (b) was not satisfied.	
3020	Restriction (c) was not satisfied.	
3600	The maximum number of iterations was reached before the specified precision was obtained.	The value at that time is returned.
4000	An error occurred in the function 3.2.16 {ASL_d1cdbt / ASL_r1cdbt}.	Processing is aborted.

(6) **Notes**

(a) The mean and variance of a beta distribution having parameters a and b are given by the following equations.

$$E[x] = \frac{a}{a+b}, \quad \sigma^2[x] = \frac{ab}{(a+b)^2(a+b+1)}$$

(b) A beta distribution with $a = b = 1$ is a uniform distribution on the interval (0, 1).

(7) **Example**

(a) Problem

For $a = 5.0$ and $b = 2.0$, obtain the values of x for which the cumulative distribution function $P(x;\alpha,\beta) = 0.7$, or $Q(x;\alpha,\beta) = 0.7$, respectively.

(b) Input data

a = 5.0, b = 2.0 and xi = 0.7.

(c) Main program

```
/*      C interface example for ASL_d1cdib */

#include <stdio.h>
#include <stdlib.h>
#include <asl.h>

int main()
{
    double a;
    double b;
    double xi;
    double xo;
    int isw;
    int ierr;


    printf( "    *** ASL_d1cdib ***\n" );
    printf( "\n     ** Input **\n\n" );

    a = 5.0;
    b = 2.0;
    xi = 0.7;
    printf( "\ta  = %8.3g\n", a );
    printf( "\tb  = %8.3g\n", b );
    printf( "\txi = %8.3g\n", xi );

    isw = 1;
    ierr = ASL_d1cdib(a, b, xi, &xo, isw);

    printf( "\n\n     ** Output **\n\n" );
    printf( "\tierr = %6d\n\n", ierr );
    printf( "\tValue of x corresponding to P(x;a,b) = xi : %8.3g\n\n", xo );

    isw = 2;
    ierr = ASL_d1cdib(a, b, xi, &xo, isw);

    printf( "\n     ** Output **\n\n" );
    printf( "\tierr = %6d\n\n", ierr );
    printf( "\tValue of x corresponding to Q(x;a,b) = xi : %8.3g\n", xo );

    return 0;
}
```

(d) Output results

```
*** ASL_d1cdib ***

 ** Input **

a  =        5
b  =        2
xi =      0.7


 ** Output **

ierr =      0

Value of x corresponding to P(x;a,b) = xi :    0.818
```

```
 ** Output **
ierr =      0
Value of x corresponding to Q(x;a,b) = xi :     0.64
```

3.2.18 ASL_d1cduf, ASL_r1cduf Uniform Distribution

(1) **Function**

For a uniform distribution within the interval (a, b), the ASL_d1cduf or ASL_r1cduf obtains the values of the following functions.

(a) probability density function; p.d.f.

$$f(x; a, b) = \begin{cases} \dfrac{1}{b-a} & (a \leq x \leq b) \\ 0 & (x < a, x > b) \end{cases}$$

(b) cumulative distribution function; c.d.f.

$$F(x; a, b) = \begin{cases} 0 & (x < a) \\ \dfrac{x-a}{b-a} & (a \leq x \leq b) \\ 1 & (x > b) \end{cases}$$

(2) **Usage**

Double precision:

```
ierr = ASL_d1cduf  (xl, xu, xi, &xo, isw);
```

Single precision:

```
ierr = ASL_r1cduf  (xl, xu, xi, &xo, isw);
```

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	xl	{D / R}	1	Input	Lower bound a of the interval for the random variable x
2	xu	{D / R}	1	Input	Upper bound b of the interval for the random variable x
3	xi	{D / R}	1	Input	Value of random variable x
4	xo	{D∗ / R∗}	1	Output	Value of the probability density function $f(x;a,b)$ or the cumulative distribution function $F(x;a,b)$ of the uniform distribution.
5	isw	I	1	Input	isw=0: Obtain the value of the probability density function $f(x;a,b)$ for xo isw=1: Obtain the value of the cumulative distribution function $F(x;a,b)$ for xo
6	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) xl≤xu

(b) isw∈{0, 1}

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
3000	Restriction (a) was not satisfied.	Processing is aborted.
3100	Restriction (b) was not satisfied.	

(6) **Notes**

None

(7) **Example**

(a) Problem

Let a=0.0, b=1.0 and x=0.5 and obtain the values of the probability density function $f(x;a,b)$ and the cumulative distribution function $F(x;a,b)$.

(b) Input data

xl=0.0, xu=1.0 and xi=0.5.

(c) Main program

```
/*      C interface example for ASL_d1cduf */
#include <stdio.h>
```

```
#include <stdlib.h>
#include <asl.h>

int main()
{
    double xl;
    double xu;
    double xi;
    double xo;
    int isw;
    int ierr;


    printf( "    *** ASL_d1cduf ***\n" );
    printf( "\n     ** Input **\n\n" );

    xl = 0.0;
    xu = 1.0;
    xi = 0.5;

    printf( "\txl = %6.3g\n", xl );
    printf( "\txu = %6.3g\n", xu );
    printf( "\txi = %6.3g\n", xi );

    printf( "\n     ** Output **\n\n" );

    isw = 0;
    ierr = ASL_d1cduf(xl, xu, xi, &xo, isw);

    printf( "\tierr = %6d\n", ierr );
    printf( "\tValue of P.D.F. = %8.3g\n\n", xo );

    isw = 1;
    ierr = ASL_d1cduf(xl, xu, xi, &xo, isw);

    printf( "\tierr = %6d\n", ierr );
    printf( "\tValue of C.D.F. = %8.3g\n", xo );

    return 0;
}
```

(d) Output results

```
*** ASL_d1cduf ***

 ** Input **

xl =      0
xu =      1
xi =    0.5

 ** Output **

ierr =      0
Value of P.D.F. =        1

ierr =      0
Value of C.D.F. =      0.5
```

3.2.19 ASL_d1cdtr, ASL_r1cdtr Triangular Distribution

(1) **Function**

For a triangular distribution, the ASL_d1cdtr or ASL_r1cdtr obtains the values of the following functions.

(a) probability density function; p.d.f.

$$f(x;a,b,c) = \begin{cases} \dfrac{2(x-a)}{(b-a)(c-a)} & (a \le x \le b) \\ \dfrac{2(c-x)}{(c-a)(c-b)} & (b < x \le c) \\ 0 & (x < a, x > c) \end{cases}$$

(b) cumulative distribution function; c.d.f.

$$F(x;a,b,c) = \begin{cases} 0 & (x < a) \\ \dfrac{(x-a)^2}{(b-a)(c-a)} & (a \le x \le b) \\ 1 - \dfrac{(c-x)^2}{(c-a)(c-b)} & (b < x \le c) \\ 1 & (x > c) \end{cases}$$

(2) **Usage**

Double precision:

```
ierr = ASL_d1cdtr  (a, b, c, xi, &xo, isw);
```

Single precision:

```
ierr = ASL_r1cdtr  (a, b, c, xi, &xo, isw);
```

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	a	{D / R}	1	Input	x coordinate of the left end of the triangular distribution. (See Notes (a))
2	b	{D / R}	1	Input	x coordinate of the apex of the triangular distribution. (See Notes (a))
3	c	{D / R}	1	Input	x coordinate of the right end of the triangular distribution. (See Notes (a))
4	xi	{D / R}	1	Input	Value of the random variable x
5	xo	{D∗ / R∗}	1	Output	Value of the probability density function $f(x; a, b, c)$ or the cumulative distribution function $F(x; a, b, c)$ of the triangular distribution.
6	isw	I	1	Input	isw=0: Obtain the value of the probability density function $f(x; a, b, c)$ for xo isw=1: Obtain the value of the cumulative distribution function $F(x; a, b, c)$ for xo
7	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) a≤b≤c

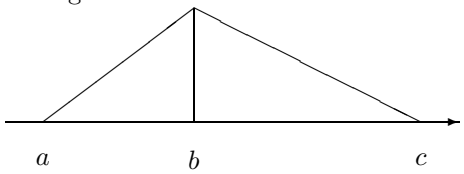
(b) isw∈{0, 1}

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
3000	Restriction (a) was not satisfied.	Processing is aborted.
3100	Restriction (b) was not satisfied.	

(6) **Notes**

(a) Triangle coordinates



a: x coordinate of the left end of the triangular distribution

b: x coordinate of the apex of the triangular distribution

c: x coordinate of the right end of the triangular distribution

(7) **Example**

(a) Problem

Let a=0.0, b=1.0, c=2.0 and x=0.5 and obtain the values of the probability density function $f(x; a, b, c)$ and the cumulative distribution function $F(x; a, b, c)$.

(b) Input data

a=0.0, b=1.0, c=2.0 and xi=0.5.

(c) Main program

```
/*      C interface example for ASL_d1cdtr */

#include <stdio.h>
#include <stdlib.h>
#include <asl.h>

int main()
{
    double a;
    double b;
    double c;
    double xi;
    double xo;
    int isw;
    int ierr;


    printf( "    *** ASL_d1cdtr ***\n" );
    printf( "\n     ** Input **\n\n" );

    a = 0.0;
    b = 1.0;
    c = 2.0;
    xi = 0.5;

    printf( "\t a = %6.3g\n", a );
    printf( "\t b = %6.3g\n", b );
    printf( "\t c = %6.3g\n", c );
    printf( "\txi = %6.3g\n", xi );

    printf( "\n     ** Output **\n\n" );

    isw = 0;
    ierr = ASL_d1cdtr(a, b, c, xi, &xo, isw);

    printf( "\tierr = %6d\n", ierr );
    printf( "\tValue of P.D.F. = %8.3g\n\n", xo );

    isw = 1;
    ierr = ASL_d1cdtr(a, b, c, xi, &xo, isw);

    printf( "\tierr = %6d\n", ierr );
    printf( "\tValue of C.D.F. = %8.3g\n", xo );

    return 0;
}
```

(d) Output results

```
*** ASL_d1cdtr ***

 ** Input **

 a =      0
 b =      1
 c =      2
xi =    0.5

 ** Output **

ierr =      0
Value of P.D.F. =      0.5

ierr =      0
Value of C.D.F. =    0.125
```

3.2.20 ASL_d1cdpa, ASL_r1cdpa Pareto Distribution

(1) **Function**

For a Pareto distribution having parameters a and b ($a > 1$, $b > 0$), the ASL_d1cdpa or ASL_r1cdpa obtains the values of the following functions.

(a) probability density function; p.d.f.

$$f(x;a,b) = \begin{cases} (a-1)(\frac{x}{b})^{-a}\frac{1}{b} & (x > b; a > 1, b > 0) \\ 0 & (x \leq b; a > 1, b > 0) \end{cases}$$

(b) cumulative distribution function; c.d.f.

$$P(x;a,b) = \int_b^x f(t;a,b)\ dt$$

$$= \begin{cases} 1 - (\frac{x}{b})^{1-a} & (x > b; a > 1, b > 0) \\ 0 & (x \leq b; a > 1, b > 0) \end{cases}$$

(c) c.d.f.

$$Q(x;a,b) = 1 - P(x;a,b)$$

$$= \begin{cases} (\frac{x}{b})^{1-a} & (x > b; a > 1, b > 0) \\ 1 & (x \leq b; a > 1, b > 0) \end{cases}$$

(2) **Usage**

Double precision:

```
ierr = ASL_d1cdpa  (a, b, xi, &xo, isw);
```

Single precision:

```
ierr = ASL_r1cdpa  (a, b, xi, &xo, isw);
```

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	a	{D / R}	1	Input	Value of parameter a.
2	b	{D / R}	1	Input	Value of parameter b.
3	xi	{D / R}	1	Input	isw=0:Value of random variable x isw=1 or 2:Integration range of random variable x
4	xo	{D∗ / R∗}	1	Output	Value of the probability density function $f(x;a,b)$ or of the cumulative distribution function $P(x;a,b)$ or $Q(x;a,b)$ of the Pareto distribution.
5	isw	I	1	Input	Processing switch isw=0:Obtain the value of the probability density function $f(x;a,b)$ for xo isw=1: Obtain the value of the cumulative distribution function $P(x;a,b)$ for xo isw=2:Obtain the value of the cumulative distribution function $Q(x;a,b)$
6	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) isw $\in \{0,1,2\}$

(b) a > 1.0, b > 0.0

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
1000	xi $\leq$ b.	When isw = 0, xo = 0.0 is set. When isw = 1, xo = 0.0 is set. When isw = 2, xo = 1.0 is set.
3000	Restriction (a) was not satisfied.	Processing is aborted.
3010	Restriction (b) was not satisfied.	

(6) **Notes**

None

(7) **Example**

(a) Problem

Let $a = 5.0, b = 2.0$, and $x = 3.0$ and obtain the values of the probability density function $f(x; a, b)$ and the cumulative distribution functions $P(x; a, b)$ and $Q(x; a, b)$.

(b) Input data

a = 5.0, b = 2.0 and x = 3.0.

(c) Main program

```
/*      C interface example for ASL_d1cdpa */

#include <stdio.h>
#include <stdlib.h>
#include <asl.h>

int main()
{
    double a;
    double b;
    double xi;
    double xo;
    int isw;
    int ierr;


    printf( "    *** ASL_d1cdpa ***\n" );
    printf( "\n     ** Input **\n\n" );

    a = 5.0;
    b = 2.0;
    xi = 3.0;

    printf( "\ta  = %8.3g\n", a );
    printf( "\tb  = %8.3g\n", b );
    printf( "\txi = %8.3g\n", xi );

    isw = 0;
    ierr = ASL_d1cdpa(a, b, xi, &xo, isw);

    printf( "\n\n     ** Output **\n\n" );
    printf( "\tierr = %6d\n\n", ierr );
    printf( "\tP.D.F = %8.3g\n\n", xo );

    isw = 1;
    ierr = ASL_d1cdpa(a, b, xi, &xo, isw);

    printf( "\n     ** Output **\n\n" );
    printf( "\tierr = %6d\n\n", ierr );
    printf( "\tC.D.F(1) = %8.3g\n\n", xo );

    isw = 2;
    ierr = ASL_d1cdpa(a, b, xi, &xo, isw);

    printf( "\n     ** Output **\n\n" );
    printf( "\tierr = %6d\n\n", ierr );
    printf( "\tC.D.F(2) = %8.3g\n\n", xo );

    return 0;
}
```

(d) Output results

```
*** ASL_d1cdpa ***

 ** Input **

a  =        5
b  =        2
xi =        3


 ** Output **

ierr =      0

P.D.F =    0.263


 ** Output **

ierr =      0

C.D.F(1) =    0.802
```

```
 ** Output **
ierr =      0
C.D.F(2) =    0.198
```

3.2.21 ASL_d1cdwe, ASL_r1cdwe Weibull Distribution

(1) **Function**

For a Weibull distribution having parameters a and b ($a > 0$, $b > 0$), the ASL_d1cdwe or ASL_r1cdwe obtains the values of the following functions.

(a) probability density function; p.d.f.

$$f(x;a,b) = \begin{cases} a\left(\frac{x}{b}\right)^{a-1} e^{-(\frac{x}{b})^a} \frac{1}{b} & (0 < x; a, b > 0) \\ 0 & (x \leq 0; a, b > 0) \end{cases}$$

(b) cumulative distribution function; c.d.f.

$$P(x;a,b) = \int_0^x f(t;a,b)\, dt$$

$$= \begin{cases} 1 - e^{-(\frac{x}{b})^a} & (0 < x; a, b > 0) \\ 0 & (x \leq 0; a, b > 0) \end{cases}$$

(c) c.d.f.

$$Q(x;a,b) = 1 - P(x;a,b)$$

$$= \begin{cases} e^{-(\frac{x}{b})^a} & (0 < x; a, b > 0) \\ 1 & (x \leq 0; a, b > 0) \end{cases}$$

(2) **Usage**

Double precision:

```
ierr = ASL_d1cdwe  (a, b, xi, &xo, isw);
```

Single precision:

```
ierr = ASL_r1cdwe  (a, b, xi, &xo, isw);
```

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	a	{D / R}	1	Input	Value of shape parameter a.
2	b	{D / R}	1	Input	Value of scale parameter b.
3	xi	{D / R}	1	Input	isw=0: Value of random variable x isw=1 or 2: Integration range of random variable x
4	xo	{D∗ / R∗}	1	Output	Value of the probability density function $f(x;a,b)$ or of the cumulative distribution function $P(x;a,b)$ or $Q(x;a,b)$ of the Weibull distribution
5	isw	I	1	Input	Processing switch isw=0:Obtain the value of the probability density function $f(x;a,b)$ for xo isw=1:Obtain the value of the cumulative distribution function $P(x;a,b)$ for xo isw=2:Obtain the value of the cumulative distribution function $Q(x;a,b)$ for xo
6	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) isw $\in \{0, 1, 2\}$

(b) a > 0.0, b > 0.0

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
1000	xi ≤ 0.0	When isw = 0, xo = 0.0 is set. When isw = 1, xo = 0.0 is set. When isw = 2, xo = 1.0 is set.
3000	Restriction (a) was not satisfied.	Processing is aborted.
3010	Restriction (b) was not satisfied.	

(6) **Notes**

None

(7) **Example**

(a) Problem

Let $a = 5.0, b = 2.0$, and $x = 2.0$ and obtain the values of the probability density function $f(x; a, b)$ and the cumulative distribution functions $P(x; a, b)$ and $Q(x; a, b)$.

(b) Input data

a = 5.0, b = 2.0 and xi = 2.0.

(c) Main program

```
/*      C interface example for ASL_d1cdwe */

#include <stdio.h>
#include <stdlib.h>
#include <asl.h>

int main()
{
    double a;
    double b;
    double xi;
    double xo;
    int isw;
    int ierr;


    printf( "    *** ASL_d1cdwe ***\n" );
    printf( "\n     ** Input **\n\n" );

    a = 5.0;
    b = 2.0;
    xi = 2.0;

    printf( "\ta  = %8.3g\n", a );
    printf( "\tb  = %8.3g\n", b );
    printf( "\txi = %8.3g\n", xi );

    isw = 0;
    ierr = ASL_d1cdwe(a, b, xi, &xo, isw);

    printf( "\n\n     ** Output **\n\n" );
    printf( "\tierr = %6d\n\n", ierr );
    printf( "\tValue of P.D.F = %8.3g\n\n", xo );

    isw = 1;
    ierr = ASL_d1cdwe(a, b, xi, &xo, isw);

    printf( "\n     ** Output **\n\n" );
    printf( "\tierr = %6d\n\n", ierr );
    printf( "\tValue of C.D.F(1) = %8.3g\n\n", xo );

    isw = 2;
    ierr = ASL_d1cdwe(a, b, xi, &xo, isw);

    printf( "\n     ** Output **\n\n" );
    printf( "\tierr = %6d\n\n", ierr );
    printf( "\tValue of C.D.F(2) = %8.3g\n\n", xo );

    return 0;
}
```

(d) Output results

```
*** ASL_d1cdwe ***

 ** Input **

a  =        5
b  =        2
xi =        2


 ** Output **

ierr =      0

Value of P.D.F =     0.92


 ** Output **

ierr =      0

Value of C.D.F(1) =    0.632
```

```
 ** Output **
ierr =      0
Value of C.D.F(2) =    0.368
```

3.2.22 ASL_d1cdex, ASL_r1cdex Exponential Distribution

(1) **Function**

For a exponential distribution having parameters λ , the ASL_d1cdex or ASL_r1cdex obtains the values of the following functions.

(a) probability density function; p.d.f.

$$f(x;\lambda) = \begin{cases} \lambda e^{-\lambda x} & (x > 0; \lambda > 0) \\ 0 & (x \leq 0; \lambda > 0) \end{cases}$$

(b) cumulative distribution function; c.d.f.

$$P(x;\lambda) = \int_0^x \lambda e^{-\lambda t} dt \quad (\lambda > 0)$$

(c) c.d.f.

$$Q(x;\lambda) = 1 - P(x;\lambda) = \int_x^\infty \lambda e^{-\lambda t} dt \quad (\lambda > 0)$$

(2) **Usage**

Double precision:

```
ierr = ASL_d1cdex  (b, xi, &xo, isw);
```

Single precision:

```
ierr = ASL_r1cdex  (b, xi, &xo, isw);
```

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	b	{D / R}	1	Input	Value of scale parameter λ
2	xi	{D / R}	1	Input	Value of random variable x
3	xo	{D∗ / R∗}	1	Output	Value of the probability density function $f(x;\lambda)$ of the gamma distribution or of the cumulative distribution function $P(x;\lambda)$ or $Q(x;\lambda)$ of the exponential distribution.
4	isw	I	1	Input	isw=0: Obtain the value of the probability density function $f(x;\lambda)$ for xo isw=1: Obtain the value of the cumulative distribution function $P(x;\lambda)$ for xo isw=2: Obtain the value of the cumulative distribution function $Q(x;\lambda)$ for xo
5	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) isw $\in \{0, 1, 2\}$

(b) b > 0.0

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
1000	xi ≤ 0.0	0.0 or 1.0 is set for xo.
3000	Restriction (a) was not satisfied.	Processing is aborted.
3010	Restriction (b) was not satisfied.	

(6) **Notes**

(a) The mean and variance of a gamma distribution having parameters λ are given by the following equations.

$$E[x] = \frac{1}{\lambda}, \sigma^2[x] = \frac{1}{\lambda^2}$$

(b) A exponential distribution is gamma distribution for which $\alpha = 1$.

(7) **Example**

(a) Problem
Let $\lambda = 2.0$ and $x = 1.0$ and obtain the values of the probability density function $f(x; \lambda)$ and the cumulative distribution functions $P(x; \lambda)$ and $Q(x; \lambda)$.

(b) Input data
b = 2.0 and xi = 1.0.

(c) Main program

```
/*      C interface example for ASL_d1cdex */

#include <stdio.h>
#include <asl.h>

int main()
{
    double b;
    double xi;
    double xo;
    int isw;
    int ierr;

    b =2.0;
    xi=1.0;

    printf( "    *** ASL_d1cdex ***\n" );
    printf( "\n     ** Input **\n\n" );
    printf( "\tb  = %8.3g \n", b );
    printf( "\txi = %8.3g \n", xi );
    printf( "\n");

    isw=0;
    ierr = ASL_d1cdex(b, xi, &xo, isw);
    printf( "\n     ** Output **\n\n" );
    printf( "\tierr = %6d\n\n", ierr );
    printf( "\tP.D.F   =%8.3g\n\n", xo );

    isw=1;
    ierr = ASL_d1cdex(b, xi, &xo, isw);
    printf( "\n     ** Output **\n\n" );
    printf( "\tierr = %6d\n\n", ierr );
    printf( "\tC.D.F(1)=%8.3g\n\n", xo );

    isw=2;
    ierr = ASL_d1cdex(b, xi, &xo, isw);
    printf( "\n     ** Output **\n\n" );
```

```
    printf( "\tierr = %6d\n\n", ierr );
    printf( "\tC.D.F(2)=%8.3g\n\n", xo );

    return 0;
}
```

(d) Output results

```
*** ASL_d1cdex ***

 ** Input **

b  =        2
xi =        1


 ** Output **

ierr =      0

P.D.F   =   0.271


 ** Output **

ierr =      0

C.D.F(1)=   0.865


 ** Output **

ierr =      0

C.D.F(2)=   0.135
```

3.2.23 ASL_d1cdgu, ASL_r1cdgu Gumbel Distribution

(1) **Function**

For a Gumbel distribution having the parameters a and b ($b > 0$), the ASL_d1cdgu or ASL_r1cdgu obtains the values of the following functions.

(a) probability density function; p.d.f.

$$f(x;a,b) = \frac{1}{b} e^{\frac{x-a}{b}} e^{-e^{\frac{x-a}{b}}}$$

(b) cumulative distribution function; c.d.f.

$$P(x;a,b) = \int_{-\infty}^{x} f(t;a,b)dt$$

(c) c.d.f.

$$Q(x;a,b) = 1 - P(x;a,b) = \int_{x}^{\infty} f(t;a,b)dt$$

(2) **Usage**

Double precision:

```
ierr = ASL_d1cdgu  (a, b, xi, &xo, isw);
```

Single precision:

```
ierr = ASL_r1cdgu  (a, b, xi, &xo, isw);
```

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	a	{D / R}	1	Input	Value of the location parameter a.
2	b	{D / R}	1	Input	Value of the scale parameter b.
3	xi	{D / R}	1	Input	isw=0:Value of random variable x isw=1 or 2:Integration range of random variable x
4	xo	{D∗ / R∗}	1	Output	Value of the probability density function $f(x;a,b)$ or of the cumulative distribution function $P(x;a,b)$ or $Q(x;a,b)$ of the Gumbel distribution
5	isw	I	1	Input	Processing switch isw=0 : Obtain the value of the probability density function $f(x;a,b)$ for xo isw=1:Obtain the value of the cumulative distribution function $P(x;a,b)$ for xo isw=2:Obtain the value of the cumulative distribution function $Q(x;a,b)$ for xo
6	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) b>0.0

(b) isw $\in \{0,1,2\}$

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
3000	Restriction (a) was not satisfied.	Processing is aborted.
3010	Restriction (b) was not satisfied.	

(6) **Notes**

None

(7) **Example**

(a) Problem

Let a=1.0, b=2.0 and x=1.5 and obtain the values of the probability density function $f(x;a,b)$ and the cumulative distribution function $P(x;a,b)$ and $Q(x;a,b)$.

(b) Input data

xl=1.0, xu=2.0 and xi=1.5.

(c) Main program

```
/*      C interface example for ASL_d1cdgu */

#include <stdio.h>
#include <stdlib.h>
#include <asl.h>

int main()
{
    double a;
    double b;
    double xi;
    double xo;
    int isw;
    int ierr;

    printf( "    *** ASL_d1cdgu ***\n" );
    printf( "\n     ** Input **\n\n" );

    a = 1.0;
    b = 2.0;
    xi = 1.5;

    printf( "\ta = %6.3g\n", a );
    printf( "\tb = %6.3g\n", b );
    printf( "\txi = %6.3g\n", xi );

    printf( "\n     ** Output **\n\n" );

    isw = 0;
    ierr = ASL_d1cdgu(a, b, xi, &xo, isw);

    printf( "\tierr = %6d\n", ierr );
    printf( "\tValue of P.D.F. = %8.3g\n\n", xo );

    isw = 1;
    ierr = ASL_d1cdgu(a, b, xi, &xo, isw);

    printf( "\tierr = %6d\n", ierr );
    printf( "\tValue of C.D.F. = %8.3g\n\n", xo );

    isw = 2;
    ierr = ASL_d1cdgu(a, b, xi, &xo, isw);

    printf( "\tierr = %6d\n", ierr );
    printf( "\tValue of C.D.F. = %8.3g\n", xo );

    return 0;
}
```

(d) Output results

```
*** ASL_d1cdgu ***

 ** Input **

a =      1
b =      2
xi =    1.5

 ** Output **

ierr =      0
Value of P.D.F. =    0.178

ierr =      0
Value of C.D.F. =    0.723

ierr =      0
Value of C.D.F. =    0.277
```

3.2.24 ASL_d1cdld, ASL_r1cdld Logarithmic Distribution

(1) **Function**

For a logarithmic distribution within the interval (a, b), the ASL_d1cdld or ASL_r1cdld obtains the values of the following functions.

(a) probability density function; p.d.f.

$$f(x; a, b) = \frac{\log x}{b(\log b - 1) - a(\log a - 1)}$$

(b) cumulative distribution function; c.d.f.

$$P(x; a, b) = \int_a^x f(t; a, b)dt$$

(c) c.d.f.

$$Q(x; a, b) = 1 - P(x; a, b) = \int_x^b f(t; a, b)dt$$

(2) **Usage**

Double precision:

```
ierr = ASL_d1cdld  (xl, xu, xi, &xo, isw);
```

Single precision:

```
ierr = ASL_r1cdld  (xl, xu, xi, &xo, isw);
```

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	xl	{D / R}	1	Input	Lower bound a of the interval for the variable x
2	xu	{D / R}	1	Input	Upper bound b of the interval for the variable x
3	xi	{D / R}	1	Input	Value of the variable x
4	xo	{D∗ / R∗}	1	Output	Value of the probability density function $f(x)$ or the cumulative distribution function $F(x)$ of the logarithmic distribution.
5	isw	I	1	Input	isw=0: Obtain the value of the probability density function $f(x)$ for xo isw=1: Obtain the value of the cumulative distribution function $P(x)$ for xo isw=2: Obtain the value of the cumulative distribution function $Q(x)$ for xo
6	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) 0.0<xl<xu

(b) xl<xi<xu

(c) isw $\in \{0, 1, 2\}$

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
3000	Restriction (a) was not satisfied.	Processing is aborted.
3010	Restriction (b) was not satisfied.	
3020	Restriction (c) was not satisfied.	

(6) **Notes**

None

(7) **Example**

(a) Problem

Let a=0.0, b=2.0 and x=1.5 and obtain the values of the probability density function $f(x;a,b)$ and the cumulative distribution function $P(x;a,b)$ and $Q(x;a,b)$.

(b) Input data

xl=1.0, xu=2.0 and xi=1.5.

(c) Main program

```
/*      C interface example for ASL_d1cdld */

#include <stdio.h>
#include <stdlib.h>
#include <asl.h>

int main()
{
    double xl;
    double xu;
    double xi;
    double xo;
    int isw;
    int ierr;

    printf( "    *** ASL_d1cdld ***\n" );
    printf( "\n     ** Input **\n\n" );

    xl = 1.0;
    xu = 2.0;
    xi = 1.5;

    printf( "\txl = %6.3g\n", xl );
    printf( "\txu = %6.3g\n", xu );
    printf( "\txi = %6.3g\n", xi );

    printf( "\n     ** Output **\n\n" );

    isw = 0;
    ierr = ASL_d1cdld(xl, xu, xi, &xo, isw);

    printf( "\tierr = %6d\n", ierr );
    printf( "\tValue of P.D.F. = %8.3g\n\n", xo );

    isw = 1;
    ierr = ASL_d1cdld(xl, xu, xi, &xo, isw);

    printf( "\tierr = %6d\n", ierr );
    printf( "\tValue of C.D.F. = %8.3g\n\n", xo );

    isw = 2;
    ierr = ASL_d1cdld(xl, xu, xi, &xo, isw);

    printf( "\tierr = %6d\n", ierr );
    printf( "\tValue of C.D.F. = %8.3g\n", xo );

    return 0;
}
```

(d) Output results

```
*** ASL_d1cdld ***

 ** Input **

xl =      1
xu =      2
xi =    1.5

 ** Output **

ierr =      0
Value of P.D.F. =     1.05

ierr =      0
Value of C.D.F. =     0.28

ierr =      0
Value of C.D.F. =     0.72
```

3.2.25 ASL_d1cdln, ASL_r1cdln Log-Normal Distribution

(1) **Function**

For a log-normal distribution having mean $e^{\mu}\sqrt{e^{\sigma^2}}$ and variance $e^{2\mu}e^{\sigma^2}(e^{\sigma^2}-1)$, the ASL_d1cdln or ASL_r1cdln obtains the values of the following functions.

(a) probability density function; p.d.f.

$$f(x;\mu,\sigma) = \frac{1}{x\sigma\sqrt{2\pi}}e^{-\frac{(\ln x-\mu)^2}{2\sigma^2}} \quad (\sigma > 0)$$

(b) cumulative distribution function; c.d.f.

$$P(x;\mu,\sigma) = \int_0^x \frac{1}{t\sigma\sqrt{2\pi}}e^{-\frac{(\ln t-\mu)^2}{2\sigma^2}}dt \quad (\sigma > 0)$$

(c) c.d.f.

$$Q(x;\mu,\sigma) = 1 - P(x;\mu,\sigma) = \int_x^\infty \frac{1}{t\sigma\sqrt{2\pi}}e^{-\frac{(\ln t-\mu)^2}{2\sigma^2}}dt \quad (\sigma > 0)$$

(2) **Usage**

Double precision:

```
ierr = ASL_d1cdln  (xe, xv, xi, &xo, isw);
```

Single precision:

```
ierr = ASL_r1cdln  (xe, xv, xi, &xo, isw);
```

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	xe	{D / R}	1	Input	Value of parameter μ
2	xv	{D / R}	1	Input	Value of parameter σ^2
3	xi	{D / R}	1	Input	Value of random variable x
4	xo	{D∗ / R∗}	1	Output	Value of the probability density function $f(x)$ of the log-normal distribution or of the cumulative distribution function $P(x;\mu,\sigma)$ or $Q(x;\mu,\sigma)$ of the normal distribution.
5	isw	I	1	Input	isw=0: Obtain the value of the probability density function $f(x;\mu,\sigma)$ for xo isw=1: Obtain the value of the cumulative distribution function $P(x;\mu,\sigma)$ for xo isw=2:Obtain the value of the cumulative distribution function $Q(x;\mu,\sigma)$ for xo
6	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) isw $\in \{0, 1, 2\}$

(b) xv > 0.0

(c) x > 0.0

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
3000	Restriction (a) was not satisfied.	Processing is aborted.
3010	Restriction (b) was not satisfied.	
3020	Restriction (c) was not satisfied.	

(6) **Notes**

(a) From the relational expression $P(x;\mu,\sigma) + Q(x;\mu,\sigma) = 1$, it is possible to obtain either $P(x;\mu,\sigma)$ or $Q(x;\mu,\sigma)$ from the other. However, there may be times when cancellation of significant digits occurs, preventing good precision from being obtained.

(b) When the random variable x obeys a log-normal distribution having mean $e^{\mu}\sqrt{e^{\sigma^2}}$ and variance $e^{2\mu}e^{\sigma^2}(e^{\sigma^2}-1)$, the random variable $\ln x$ obeys a normal distribution $N(\mu,\sigma^2)$.

(7) **Example**

(a) Problem
Let $\mu = 5.0$, $\sigma^2 = 2.5$ and $x = 3.0$ and obtain the values of the probability density function $f(x;\mu,\sigma)$ and the cumulative distribution functions $P(x;\mu,\sigma)$ and $Q(x;\mu,\sigma)$.

(b) Input data
xe = 5.0, xv = 2.5 and xi = 3.0.

(c) Main program

```
/*      C interface example for ASL_d1cdln */

#include <stdio.h>
#include <asl.h>

int main()
{
    double xe;
    double xv;
    double xi;
    double xo;
    int isw;
    int ierr;

    xe=5.0;
    xv=2.5;
    xi=20.08553692318766;

    printf( "    *** ASL_d1cdln ***\n" );
    printf( "\n     ** Input **\n\n" );

    printf( "\txe = %8.3g xv = %8.3g xi = %8.3g\n", xe, xv, xi );

    printf( "\n     ** Output **\n\n" );
    isw=0;
    ierr = ASL_d1cdln(xe, xv, xi, &xo, isw);

    printf( "\tierr = %6d\n", ierr );
    printf( "\tValue of P.D.F   =%8.3g\n\n", xo );
    isw=1;
    ierr = ASL_d1cdln(xe, xv, xi, &xo, isw);

    printf( "\tierr = %6d\n", ierr );
    printf( "\tValue of C.D.F(1)=%8.3g\n\n", xo );
    isw=2;
    ierr = ASL_d1cdln(xe, xv, xi, &xo, isw);

    printf( "\tierr = %6d\n", ierr );
    printf( "\tValue of C.D.F(2)=%8.3g\n\n", xo );

    return 0;
}
```

(d) Output results

```
*** ASL_d1cdln ***

 ** Input **

xe =        5 xv =      2.5 xi =     20.1

 ** Output **

ierr =      0
Value of P.D.F   = 0.00564

ierr =      0
Value of C.D.F(1)=   0.103

ierr =      0
Value of C.D.F(2)=   0.897
```

3.2.26 ASL_d1cdlg, ASL_r1cdlg Logistic Distribution

(1) **Function**

For a logistic distribution having mean α and variance $\left(\sigma^2 = \frac{\pi^2\beta^2}{3}\right)$ the ASL_d1cdlg or ASL_r1cdlg obtains the values of the following functions.

(a) probability density function; p.d.f.

$$f(x;\alpha,\beta) = \frac{e^{-\frac{x-\alpha}{\beta}}}{\beta\left\{1+e^{-\frac{x-\alpha}{\beta}}\right\}^2} \quad (\beta > 0)$$

$$\sigma^2 = \frac{\pi^2\beta^2}{3}$$

(b) cumulative distribution function; c.d.f.

$$P(x;\alpha,\beta) = \frac{1}{1+e^{-\frac{x-\alpha}{\beta}}} \quad (\beta > 0)$$

(c) c.d.f.

$$Q(x;\alpha,\beta) = 1 - P(x;\alpha,\beta) = \frac{1}{1+e^{\frac{x-\alpha}{\beta}}} \quad (\beta > 0)$$

(2) **Usage**

Double precision:

```
ierr = ASL_d1cdlg  (xa, xb, xi, &xo, isw);
```

Single precision:

```
ierr = ASL_r1cdlg  (xa, xb, xi, &xo, isw);
```

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	xa	{D / R}	1	Input	Value of mean α
2	xb	{D / R}	1	Input	Value of parameter β
3	xi	{D / R}	1	Input	Value of random variable x
4	xo	{D∗ / R∗}	1	Output	Value of the probability density function $f(x)$ of the logistic distribution or of the cumulative distribution function $P(x;\alpha,\beta)$ or $Q(x;\alpha,\beta)$ of the logistic distribution.
5	isw	I	1	Input	isw=0: Obtain the value of the probability density function $f(x;\alpha,\beta)$ for xo isw=1:Obtain the value of the cumulative distribution function $P(x;\alpha,\beta)$ isw=2:Obtain the value of the cumulative distribution function $Q(x;\alpha,\beta)$
6	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) isw $\in \{0,1,2\}$

(b) xb > 0.0

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
3000	Restriction (a) was not satisfied.	Processing is aborted.
3010	Restriction (b) was not satisfied.	

(6) **Notes**

(a) From the relational expression $P(x;\alpha,\beta)+Q(x;\alpha,\beta)=1$, it is possible to obtain either $P(x;\alpha,\beta)$ or $Q(x;\alpha,\beta)$ from the other. However, there may be times when cancellation of significant digits occurs, preventing good precision from being obtained.

(7) **Example**

(a) Problem

Let $\alpha = 1.0$, $\beta = 1.0$ and $x = 3.0$ and obtain the values of the probability density function $f(x;\alpha,\beta)$ and the cumulative distribution functions $P(x;\alpha,\beta)$ and $Q(x;\alpha,\beta)$.

(b) Input data

xa = 1.0, xb = 1.0 and xi = 3.0.

(c) Main program

```
/*      C interface example for ASL_d1cdlg */

#include <stdio.h>
#include <asl.h>

int main()
{
    double xa;
    double xb;
    double xi;
    double xo;
    int isw;
    int ierr;

    xa=1.0;
    xb=1.0;
    xi=3.0;

    printf( "    *** ASL_d1cdlg ***\n" );
    printf( "\n     ** Input **\n\n" );

    printf( "\txa = %8.3g xb = %8.3g xi = %8.3g\n", xa, xb, xi );

    printf( "\n     ** Output **\n\n" );
    isw=0;
    ierr = ASL_d1cdlg(xa, xb, xi, &xo, isw);

    printf( "\tierr = %6d\n", ierr );
    printf( "\tValue of P.D.F   =%8.3g\n\n", xo );
    isw=1;
    ierr = ASL_d1cdlg(xa, xb, xi, &xo, isw);

    printf( "\tierr = %6d\n", ierr );
    printf( "\tValue of C.D.F(1)=%8.3g\n\n", xo );
    isw=2;
    ierr = ASL_d1cdlg(xa, xb, xi, &xo, isw);

    printf( "\tierr = %6d\n", ierr );
    printf( "\tValue of C.D.F(2)=%8.3g\n\n", xo );

    return 0;
}
```

(d) Output results

```
*** ASL_d1cdlg ***

 ** Input **

xa =        1 xb =        1 xi =        3

 ** Output **

ierr =      0
Value of P.D.F   =   0.105

ierr =      0
Value of C.D.F(1)=   0.881

ierr =      0
Value of C.D.F(2)=   0.119
```

3.2.27 ASL_d1cdcc, ASL_r1cdcc Cauchy Distribution

(1) **Function**

For Cauchy distribution having parameters α and β, the ASL_d1cdcc or ASL_r1cdcc obtains the values of the following functions.

(a) probability density function; p.d.f.

$$f(x;\alpha,\beta) = \frac{1}{\pi}\left[\frac{\beta}{\beta^2+(x-\alpha)^2}\right] \quad (\beta > 0)$$

(b) cumulative distribution function; c.d.f.

$$\begin{aligned} P(x;\alpha,\beta) &= \int_{-\infty}^{x} f(t;\alpha,\beta)\, dt \\ &= \frac{1}{2} + \frac{1}{\pi}\tan^{-1}\frac{(x-\alpha)}{\beta} \quad (\beta > 0) \end{aligned}$$

(c) c.d.f.

$$\begin{aligned} Q(x;\alpha,\beta) &= 1 - P(x;\alpha,\beta) \\ &= \frac{1}{2} - \frac{1}{\pi}\tan^{-1}\frac{(x-\alpha)}{\beta} \quad (\beta > 0) \end{aligned}$$

(2) **Usage**

Double precision:

```
ierr = ASL_d1cdcc  (a, b, xi, &xo, isw);
```

Single precision:

```
ierr = ASL_r1cdcc  (a, b, xi, &xo, isw);
```

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	a	{D / R}	1	Input	Value of parameter α
2	b	{D / R}	1	Input	Value of parameter β
3	xi	{D / R}	1	Input	Value of random variable x
4	xo	{D∗ / R∗}	1	Output	Value of the probability density function $f(x;\alpha,\beta)$ of Cauchy distribution or of the cumulative distribution function $P(x;\alpha,\beta)$ or $Q(x;\alpha,\beta)$ of Cauchy distribution.
5	isw	I	1	Input	isw=0: Obtain the value of the probability density function $f(x;\alpha,\beta)$ for xo isw=1: Obtain the value of the cumulative distribution function $P(x;\alpha,\beta)$ for xo isw=2: Obtain the value of the cumulative distribution function $Q(x;\alpha,\beta)$ for xo
6	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) isw $\in \{0, 1, 2\}$

(b) b > 0.0

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
3000	Restriction (a) was not satisfied.	Processing is aborted.
3010	Restriction (b) was not satisfied.	

(6) **Notes**

None

(7) **Example**

(a) Problem

Let $\alpha = 2.0$, $\beta = 1.0$ and $x = 3.0$ and obtain the values of the probability density function $f(x; \alpha, \beta)$ and the cumulative distribution functions $P(x; \alpha, \beta)$ and $Q(x; \alpha, \beta)$.

(b) Input data

a = 2.0, b = 1.0 and xi = 3.0.

(c) Main program

```
/*      C interface example for ASL_d1cdcc */

#include <stdio.h>
#include <asl.h>

int main()
{
    double a;
    double b;
    double xi;
    double xo;
    int isw;
    int ierr;

    a =2.0;
    b =1.0;
    xi=3.0;

    printf( "    *** ASL_d1cdcc ***\n" );
    printf( "\n     ** Input **\n\n" );

    printf( "\ta  = %8.3g \n", a );
    printf( "\tb  = %8.3g \n", b );
    printf( "\txi = %8.3g \n", xi );
    printf( "\n" );

    isw=0;
    ierr = ASL_d1cdcc(a, b, xi, &xo, isw);

    printf( "\n     ** Output **\n\n" );
    printf( "\tierr = %6d\n\n", ierr );
    printf( "\tP.D.F   =%8.3g\n\n", xo );

    isw=1;
    ierr = ASL_d1cdcc(a, b, xi, &xo, isw);

    printf( "\n     ** Output **\n\n" );
    printf( "\tierr = %6d\n\n", ierr );
    printf( "\tC.D.F(1) =%8.3g\n\n", xo );

    isw=2;
    ierr = ASL_d1cdcc(a, b, xi, &xo, isw);

    printf( "\n     ** Output **\n\n" );
    printf( "\tierr = %6d\n\n", ierr );
    printf( "\tC.D.F(2) =%8.3g\n\n", xo );

    return 0;
}
```

(d) Output results

```
*** ASL_d1cdcc ***

 ** Input **

a  =        2
b  =        1
xi =        3


 ** Output **

ierr =      0

P.D.F   =   0.159


 ** Output **

ierr =      0

C.D.F(1) =    0.75
```

```
 ** Output **
ierr =      0
C.D.F(2) =    0.25
```

3.3 DISCRETE DISTRIBUTIONS

3.3.1 ASL_d1ddbp, ASL_r1ddbp
Binomial Distribution and Negative Binomial Distribution

(1) **Function**

(a) Binomial Distribution (1)

Given the probability that an event will occur p, the number of trials n, and the number of occurrences m, the ASL_d1ddbp or ASL_r1ddbp obtains the values of the binomial distribution probability $P_{BIN}(X = m; p, n)$ and the cumulative distribution function (c.d.f.) $P_{BIN}(X \leq m; p, n)$ in m occurrences, which are defined by the following equations.

$$P_{BIN}(X = m; p, n) = \left(\begin{array}{c} n \\ m \end{array} \right) p^m \cdot q^{n-m} \quad (q = 1 - p)$$

$$P_{BIN}(X \leq m; p, n) = \sum_{i=m}^{n} \left(\begin{array}{c} n \\ i \end{array} \right) p^i \cdot q^{n-i}$$

(b) Binomial Distribution (2)

Given the probability of success in one trial p, the number of trials of independent events n, and the maximum value for the number of failures m, the ASL_d1ddbp or ASL_r1ddbp obtains the value of the probability $Q_{BIN}(X = m; p, n)$ of at least $(n - m)$ successes in n trials, which is defined by the following equation.

$$\begin{aligned} Q_{BIN}(X = m; p, n) &= P_{BIN}(X \geq n - m; p, n) \\ &= 1 - \sum_{r=0}^{n-m-1} \left(\begin{array}{c} n \\ r \end{array} \right) p^r \cdot q^{n-r} \quad (q = 1 - p) \end{aligned}$$

(c) Negative Binomial Distribution

Given the probability of success in one trial p and the number of successes n and number of failures m in repeated trials, the ASL_d1ddbp or ASL_r1ddbp obtains the values of the negative binomial distribution probability $P_{NB}(X = m; p, n)$ and cumulative distribution function (c.d.f.) $P_{NB}(X \leq m; p, n)$ in m failures, which are defined by the following equations.

$$P_{NB}(X = m; p, n) = \left(\begin{array}{c} n + m - 1 \\ m \end{array} \right) p^n \cdot q^m \qquad (q = 1 - p)$$

$$P_{NB}(X \leq m; p, n) = \sum_{i=0}^{m} \left(\begin{array}{c} n + i - 1 \\ i \end{array} \right) p^n \cdot q^i$$

(2) **Usage**

Double precision:

```
ierr = ASL_d1ddbp  (n, m, pi, &po, isw);
```

Single precision:

```
ierr = ASL_r1ddbp  (n, m, pi, &po, isw);
```

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	n	I	1	Input	Variable n
2	m	I	1	Input	Variable m
3	pi	{D / R}	1	Input	Probability p
4	po	{D∗ / R∗}	1	Output	Value of binomial distribution probability $P_{BIN}(X = m; p, n)$, binomial distribution cumulative distribution function $P_{BIN}(X \leq m; p, n)$, negative binomial distribution probability $P_{NB}(X = m; p, n)$, negative binomial distribution cumulative distribution function $P_{NB}(X \leq m; p, n)$ or probability $Q_{BIN}(X = m; p, n)$
5	isw	I	1	Input	Processing switch isw=1: Obtain the value of the binomial distribution probability $P_{BIN}(X = m; p, n)$ for po isw=2: Obtain the value of the binomial distribution cumulative distribution function $P_{BIN}(X \leq m; p, n)$ for po isw=3: Obtain the value of the negative binomial distribution probability $P_{NB}(X = m; p, n)$ for po isw=4: Obtain the value of the negative binomial distribution cumulative distribution function $P_{NB}(X \leq m; p, n)$ for po isw=5: Obtain the value of the probability $Q_{BIN}(X = m; p, n)$ for po
6	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) 0.0<pi<1.0

(b) n≥1

(c) 0≤m≤n (when isw∈{1, 2, 5})
m≥0 (when isw∈{3, 4})

(d) isw∈{1, 2, 3, 4, 5}

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
3000	Restriction (a) was not satisfied.	Processing is aborted.
3100	Restriction (b) was not satisfied.	
3200	Restriction (c) was not satisfied.	
3300	Restriction (d) was not satisfied.	

(6) **Notes**

None

(7) **Example**

(a) Problem

- Let the probability that an event will occur p=0.3, the number of trials n=10 and the number of occurrences m=3 and obtain the values of the binomial distribution probability $P_{BIN}(X = m; p, n)$ and cumulative distribution function $P_{BIN}(X \leq m; p, n)$.
- Let the probability of success in one trial p=0.3, the number of successes in repeated trials n=10 and the number of failures in repeated trials m=3 and obtain the values of the negative binomial distribution probability $P_{NB}(X = m; p, n)$ and cumulative distribution function $P_{NB}(X \leq m; p, n)$.
- Let the probability of success in one trial p=0.3, the number of trials of independent events n=10 and the maximum value of the number of failures m=3 and obtain the probability $Q_{BIN}(X = m; p, n)$.

(b) Input data

pi=0.3, n=10 and m=3.

(c) Main program

```
/*      C interface example for ASL_d1ddbp */

#include <stdio.h>
#include <stdlib.h>
#include <asl.h>

int main()
{
    int n;
    int m;
    double pi;
    double po;
    int isw;
    int ierr;


    n = 10;
    m = 3;
    pi = 0.3;

    printf( "    *** ASL_d1ddbp ***\n" );
    printf( "\n     ** Input **\n\n" );

    printf( "\tn  = %6d\n", n );
    printf( "\tm  = %6d\n", m );
    printf( "\tpi = %8.3g\n", pi );

    printf( "\n     ** Output **\n\n" );

    isw = 1;
    ierr = ASL_d1ddbp(n, m, pi, &po, isw);
    printf( "\tierr = %6d\n", ierr );
    printf( "\tValue of P.D.F. of Binomial Distribution" );
    printf( "          = %8.3g\n\n", po );

    isw = 2;
    ierr = ASL_d1ddbp(n, m, pi, &po, isw);
```

```
    printf( "\tierr = %6d\n", ierr );
    printf( "\tValue of C.D.F. of" );
    printf( " Binomial Distribution          = %8.3g\n\n", po );

    isw = 3;
    ierr = ASL_d1ddbp(n, m, pi, &po, isw);
    printf( "\tierr = %6d\n", ierr );
    printf( "\tValue of P.D.F. of" );
    printf( " Negative Binomial Distribution = %8.3g\n\n", po );

    isw = 4;
    ierr = ASL_d1ddbp(n, m, pi, &po, isw);
    printf( "\tierr = %6d\n", ierr );
    printf( "\tValue of C.D.F. of" );
    printf( " Negative Binomial Distribution = %8.3g\n\n", po );

    isw = 5;
    ierr = ASL_d1ddbp(n, m, pi, &po, isw);
    printf( "\tierr = %6d\n", ierr );
    printf( "\tValue of" );
    printf( " Binomial Probability                    = %8.3g\n", po );

    return 0;
}
```

(d) Output results

```
*** ASL_d1ddbp ***

 ** Input **

n  =     10
m  =      3
pi =      0.3

 ** Output **

ierr =      0
Value of P.D.F. of Binomial Distribution          =    0.267

ierr =      0
Value of C.D.F. of Binomial Distribution          =    0.617

ierr =      0
Value of P.D.F. of Negative Binomial Distribution = 0.000446

ierr =      0
Value of C.D.F. of Negative Binomial Distribution = 0.000652

ierr =      0
Value of Binomial Probability                     =   0.0106
```

3.3.2 ASL_d1ddgo, ASL_r1ddgo Geometric Distribution

(1) **Function**

Given the probability of success in m trial p, the ASL_d1ddbp or ASL_r1ddbp obtains the values of the geometric distribution probability $P_{NB}(X = m; p)$ and cumulative distribution function (c.d.f.) $P_{NB}(X \leq m; p)$, which are defined by the following equations.

$$P_{NB}(X = m; p) = q^{m-1}p \qquad (q = 1 - p)$$

$$P_{NB}(X \leq m; p) = \sum_{i=0}^{m} q^{m-1}p$$

(2) **Usage**

Double precision:

```
ierr = ASL_d1ddgo  (m, pi, &po, isw);
```

Single precision:

```
ierr = ASL_r1ddgo  (m, pi, &po, isw);
```

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	m	I	1	Input	Variable m
2	pi	{ D / R }	1	Input	Probability p
3	po	{ D∗ / R∗ }	1	Output	Value of geometric distribution probability $P_{NB}(X = m; p)$, geometric distribution cumulative distribution function $P_{NB}(X \leq m; p)$
4	isw	I	1	Input	Processing switch isw=1: Obtain the value of the geometric distribution probability $P_{BIN}(X = m; p)$ for po isw=2: Obtain the value of the geometric distribution cumulative distribution function $P_{BIN}(X \leq m; p)$ for po
5	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) 0.0<pi<1.0

(b) m≥0

(c) isw∈{1, 2}

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
3000	Restriction (a) was not satisfied.	Processing is aborted.
3200	Restriction (b) was not satisfied.	
3300	Restriction (c) was not satisfied.	

(6) **Notes**

None

(7) **Example**

(a) Problem

Let the probability that an event will occur p=0.3 and the number of trials m=3 and obtain the values of the binomial distribution probability $P_{BIN}(X = m; p)$ and cumulative distribution function $P_{BIN}(X \leq m; p)$.

(b) Input data

pi=0.3 and m=3.

(c) Main program

```
/*      C interface example for ASL_d1ddgo */

#include <stdio.h>
#include <stdlib.h>
#include <asl.h>

int main()
{
    int m;
    double pi;
    double po;
    int isw;
    int ierr;

    m  = 3;
    pi = 0.3;
    printf( "    *** ASL_d1ddgo ***\n" );
    printf( "\n     ** Input **\n\n"   );

    printf( "\tm  = %6d\n", m          );
    printf( "\tpi = %8.3g\n", pi       );

    printf( "\n     ** Output **\n\n" );

        isw = 1;
    ierr = ASL_d1ddgo(m, pi, &po, isw);
    printf( "\tierr = %6d\n", ierr );
    printf( "\tValue of P.D.F. of Geometric Distribution" );
    printf( "          = %8.3g\n\n", po );

        isw = 2;
    ierr = ASL_d1ddgo(m, pi, &po, isw);
    printf( "\tierr = %6d\n", ierr );
    printf( "\tValue of C.D.F. of" );
    printf( " Geometric Distribution          = %8.3g\n\n", po );

    return 0;
}
```

(d) Output results

```
*** ASL_d1ddgo ***

 ** Input **

m  =      3
pi =      0.3

 ** Output **

ierr =      0
Value of P.D.F. of Geometric Distribution          =    0.103

ierr =      0
Value of C.D.F. of Geometric Distribution          =     0.76
```

3.3.3 ASL_d1ddpo, ASL_r1ddpo Poisson Distribution

(1) **Function**

Given the mean λ and random variable k, the ASL_d1ddpo or ASL_r1ddpo obtains the value of the probability $Pr.\{X = k\}$ or cumulative distribution function $F(k)$ of a Poisson distribution, which are defined by the following expressions.

$$Pr.\{X = k\} = e^{-\lambda}\frac{\lambda^k}{k!} \qquad (k = 0, 1, 2, \cdots; \lambda > 0)$$

$$F(k) = \sum_{i=0}^{k} Pr.\{X = i\} = e^{-\lambda}\sum_{i=0}^{k}\frac{\lambda^k}{k!}$$

(2) **Usage**

Double precision:

```
ierr = ASL_d1ddpo  (xe, k, &xo, isw);
```

Single precision:

```
ierr = ASL_r1ddpo  (xe, k, &xo, isw);
```

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	xe	{D / R}	1	Input	Mean λ.
2	k	I	1	Input	Value of random variable k.
3	xo	{D∗ / R∗}	1	Output	Value of the probability $Pr.\{X = k\}$ or cumulative distribution function $F(k)$ of the Poisson distribution.
4	isw	I	1	Input	Processing switch isw=1:Obtain the value of the probability $Pr.\{X = k\}$ of the Poisson distribution for xo isw=2:Obtain the value of the cumulative distribution function $F(k)$ of the Poisson distribution for xo
5	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) isw∈{1, 2}

(b) 0.0<xe

(c) 0≤k

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
3000	Restriction (a) was not satisfied.	Processing is aborted.
3010	Restriction (b) or (c) was not satisfied.	

(6) **Notes**

(a) When k > 1000, the cumulative distribution function $F(k)$ is obtained according to the Peizer-Pratt approximation.

(7) **Example**

(a) Problem
Let the mean $\lambda = 3.0$ and the random variable $k = 2$ and obtain the values of the probability $Pr.\{X = k\}$ and cumulative distribution function $F(k)$ of the Poisson distribution.

(b) Input data
xe=3.0 and k=2.

(c) Main program

```
/*      C interface example for ASL_d1ddpo */

#include <stdio.h>
#include <stdlib.h>
#include <asl.h>

int main()
{
    double xe;
    int k;
    double xo;
    int isw;
    int ierr;


    printf( "    *** ASL_d1ddpo ***\n" );
    printf( "\n     ** Input **\n\n" );

    xe = 3.0;
    k = 2;

    printf( "\txe = %8.3g\n", xe );
    printf( "\tk = %6d\n", k );

    isw = 1;
    ierr = ASL_d1ddpo(xe, k, &xo, isw);

    printf( "\n\n     ** Output **\n\n" );
    printf( "\tierr = %6d\n\n", ierr );
    printf( "\tValue of P.D.F = %8.3g\n\n", xo );

    isw = 2;
    ierr = ASL_d1ddpo(xe, k, &xo, isw);

    printf( "\n     ** Output **\n\n" );
    printf( "\tierr = %6d\n\n", ierr );
    printf( "\tValue of C.D.F = %8.3g\n\n", xo );

    return 0;
}
```

(d) Output results

```
*** ASL_d1ddpo ***

 ** Input **

xe =        3
k =      2

 ** Output **

ierr =      0

Value of P.D.F =    0.224

 ** Output **

ierr =      0

Value of C.D.F =    0.423
```

3.3.4 ASL_d1ddhg, ASL_r1ddhg Hypergeometric Distribution

(1) **Function**

Assume that there is a lot of size N in which M of the N articles are inferior goods and $N - M$ articles are of good quality. The ASL_d1ddhg or ASL_r1ddhg obtains the values of the hypergeometric distribution probability $Pr.\{X = k\}$ and cumulative distribution function $F(k)$ corresponding to the probability distribution in which k inferior goods appear when an arbitrary sample of size n is extracted from this lot. The hypergeometric distribution probability $Pr.\{X = k\}$ and cumulative distribution function $F(k)$ are defined by the following equations.

$$Pr.\{X = k\} = \begin{cases} \dfrac{\begin{pmatrix} M \\ k \end{pmatrix}\begin{pmatrix} N-M \\ n-k \end{pmatrix}}{\begin{pmatrix} N \\ n \end{pmatrix}} & k = 0, 1, 2, \cdots, \min\{M, n\} \\ 0 & \text{otherwise} \end{cases}$$

$$F(k) = \sum_{i=0}^{k} Pr.\{X = i\} = \frac{\sum_{i=0}^{k}\begin{pmatrix} M \\ i \end{pmatrix}\begin{pmatrix} N-M \\ n-i \end{pmatrix}}{\begin{pmatrix} N \\ n \end{pmatrix}}$$

(2) **Usage**

Double precision:

```
ierr = ASL_d1ddhg  (nn, m, n, k, &xo, isw);
```

Single precision:

```
ierr = ASL_r1ddhg  (nn, m, n, k, &xo, isw);
```

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	nn	I	1	Input	Size of population N
2	m	I	1	Input	Number of inferior goods in population M
3	n	I	1	Input	Size of sample n
4	k	I	1	Input	Number of inferior goods in sample k
5	xo	{D∗ / R∗}	1	Output	Value of hypergeometric distribution probability $Pr.\{X = k\}$ or cumulative distribution function $F(k)$
6	isw	I	1	Input	isw=0: Obtain $Pr.\{X = k\}$ isw=1: Obtain $F(k)$
7	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) isw $\in \{0, 1\}$

(b) $0 \leq$ m $\leq$ nn

(c) $1 \leq$ n $\leq$ nn

(d) $0 \leq$ k $\leq \min($m, n$)$

(e) m + n − nn $\leq$ k

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
1000	Restriction (d) was not satisfied.	0.0 is set for xo. (When isw = 0 or k < 0.) 1.0 is set for xo. (When min(m, n) < k.)
1010	Restriction (e) was not satisfied.	0.0 is set for xo.
3000	Restriction (a) was not satisfied.	Processing is aborted.
3010	Restriction (b) was not satisfied.	
3020	Restriction (c) was not satisfied.	

(6) **Notes**

(a) The mathematical expectation and variance of the hypergeometric distribution are given by the following equations.

$$E(X) = np,\ \sigma^2(X) = \frac{N-k}{N-1}np(1-p)\ \ (p = \frac{\mathrm{M}}{\mathrm{N}})$$

(b) Since the hypergeometric distribution probability $Pr.\{X = k\}$ satisfies the following relationship

$$Pr.\{X = k\} = \frac{\left(\begin{array}{c} M \\ k \end{array}\right)\left(\begin{array}{c} N-M \\ n-k \end{array}\right)}{\left(\begin{array}{c} N \\ n \end{array}\right)} \simeq \left(\begin{array}{c} n \\ k \end{array}\right) p^k(1-p)^k,\ (p = \frac{M}{N}, \frac{n}{N} \ll 1)$$

it can be approximated by a binomial distribution $B(n, p)$ when the population is sufficiently large.

(7) **Example**

(a) Problem

Let $N = 1000$, $M = 50$, $n = 20$ and $k = 1$ and obtain the values of the hypergeometric distribution probability $Pr.\{X = k\}$ and cumulative distribution function $F(k)$.

(b) Input data

nn = 1000, m = 20, n = 20 and k = 1.

(c) Main program

```
/*      C interface example for ASL_d1ddhg */

#include <stdio.h>
#include <asl.h>

int main()
{
    int nn;
    int m;
    int n;
    int k;
```

```
    double xo;
    int isw;
    int ierr;


    nn=1000;
    m=50;
    n=20;
    k=1;

    printf( "    *** ASL_d1ddhg ***\n" );
    printf( "\n     ** Input **\n\n" );

    printf( "\tnn = %6d m = %6d n = %6d k = %6d\n", nn, m, n, k );

    printf( "\n     ** Output **\n\n" );
    isw=0;
    ierr = ASL_d1ddhg(nn, m, n, k, &xo, isw);

    printf( "\tierr = %6d\n", ierr );
    printf( "\tValue of P.D.F=%8.3g\n\n", xo );
    isw=1;
    ierr = ASL_d1ddhg(nn, m, n, k, &xo, isw);

    printf( "\tierr = %6d\n", ierr );
    printf( "\tValue of C.D.F=%8.3g\n", xo );

    return 0;
}
```

(d) Output results

```
*** ASL_d1ddhg ***

 ** Input **

nn =   1000 m =     50 n =     20 k =      1

 ** Output **

ierr =      0
Value of P.D.F=   0.381

ierr =      0
Value of C.D.F=   0.736
```

3.3.5 ASL_d1ddhn, ASL_r1ddhn
Negative Hypergeometric Distribution

(1) **Function**

Assume that there is a lot of size NN in which M of the NN articles are inferior goods and $NN-M$ articles are of good quality. Sampling from this lot is continued until n inferior goods are extracted. The negative hypergeometric distribution probability $Pr.\{X=k\}$ is defined as the probability of such occurrences that exactly k goods has been extracted at this time. The ASL_d1ddhn or ASL_r1ddhn obtains the values of the negative hypergeometric distribution probability $Pr.\{X=k\}$ and cumulative distribution function $F(k)$ corresponding to the probability distribution in which exactly k goods have been extracted from this lot when the n-th inferior good appears. The negative hypergeometric distribution probability $Pr.\{X=k\}$ and cumulative distribution function $F(k)$ are defined by the following equations.

$$Pr.\{X=k\} = \begin{cases} \dfrac{\begin{pmatrix} M \\ NN-1 \end{pmatrix}\begin{pmatrix} NN-M \\ k-n \end{pmatrix}}{\begin{pmatrix} NN \\ k-1 \end{pmatrix}} \times \dfrac{M-n+1}{NN-k+1} \\ = \dfrac{\begin{pmatrix} k-1 \\ n-1 \end{pmatrix}\begin{pmatrix} NN-k \\ M-n \end{pmatrix}}{\begin{pmatrix} NN \\ M \end{pmatrix}} & \text{when } k=n, n+1, n+2, \cdots, NN-M+n \\ 0 & \text{Otherwise.} \end{cases}$$

$$F(k) = \sum_{i=n}^{k} Pr.\{X=i\} = \frac{\sum_{i=0}^{k}\begin{pmatrix} i-1 \\ n-1 \end{pmatrix}\begin{pmatrix} NN-i \\ M-n \end{pmatrix}}{\begin{pmatrix} NN \\ M \end{pmatrix}}$$

(2) **Usage**

Double precision:

```
ierr = ASL_d1ddhn  (nn, m, n, k, &xo, isw);
```

Single precision:

```
ierr = ASL_r1ddhn  (nn, m, n, k, &xo, isw);
```

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	nn	I	1	Input	Size NN of population
2	m	I	1	Input	Number of inferior goods M in population
3	n	I	1	Input	Number n of inferior goods extracted from the lot
4	k	I	1	Input	Total number k of goods extracted from the lot
5	xo	{D∗ / R∗}	1	Output	Value of negative hypergeometric distribution probability $Pr.\{X = k\}$ or cumulative distribution function $F(k)$
6	isw	I	1	Input	isw=0: Obtain $Pr.\{X = k\}$ isw=1: Obtain $F(k)$
7	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) isw $\in \{0, 1\}$

(b) $1 \leq$ n $\leq$ m $\leq$ nn

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
1000	k < n	0.0 is set for xo.
1010	k > nn − m + n	If isw=0 then 0.0 is set for xo. If isw=1 then 1.0 is set for xo.
3000	Restriction (a) was not satisfied.	Processing is aborted.
3010	Restriction (b) was not satisfied.	

(6) **Notes**

(a) If $M = 1$ and if $n = 1$:
$Pr.\{X = k\} = \frac{1}{NN} \quad k = 1, 2, \cdots, NN$
When $n \neq 1$: $Pr.\{X = k\} = 0$.

(b) If $M > 1$, define $N1$ as the greatest integer no more than

$$\frac{NN \times (n-1)}{M-1} + 1.$$

Then the following stands for the distribution probability $Pr.\{X = k\}$:

$$Pr.\{X = n\} \leq Pr.\{X = n+1\} \leq Pr.\{X = n+2\} \leq \cdots \leq Pr.\{X = N1\}.$$

$$Pr.\{X = N1\} \geq Pr.\{X = N1+1\} \geq Pr.\{X = N1+2\} \geq \cdots \geq Pr.\{X = NN - M + n\}.$$

(7) **Example**

(a) Problem

Let $N = 30$, $M = 25$ and $n = 10$ and obtain the values of the hypergeometric distribution probability $Pr.\{X = k\}$ and cumulative distribution function $F(k)$ for $k = 10, 11, \cdots, 15$.

(b) Input data

nn = 30, m = 25 and n = 10.

(c) Main program

```
/*      C interface example for ASL_d1ddhn */

#include <stdio.h>
#include <stdlib.h>
#include <asl.h>

int main()
{
    int nn;
    int m;
    int n;
    int k;
    int isw;
    double xo;
    int ierr;

    printf( "    *** ASL_d1ddhn ***\n" );
    printf( "\n     ** Input **\n\n" );

    nn = 30;
    m = 25;
    n = 10;

    printf( "\tnn = %6d\n", nn );
    printf( "\tm  = %6d\n", m );
    printf( "\tn  = %6d\n", n );

    printf( "\n     ** Output **\n\n" );

    for( k=n ; k<=nn-m+n ; k++) {
        printf( "\t** k= %4d **\n",k );
        isw=0;
        ierr = ASL_d1ddhn(nn, m, n, k, &xo, isw);
        printf( "\tierr = %6d\n", ierr );
        printf( "\tValue of P.D.F. = %8.3g\n", xo );

        isw=1;
        ierr = ASL_d1ddhn(nn, m, n, k, &xo, isw);
        printf( "\tierr = %6d\n", ierr );
        printf( "\tValue of C.D.F. = %8.3g\n\n", xo );
    }

    return 0;
}
```

(d) Output results

```
*** ASL_d1ddhn ***

 ** Input **

nn =     30
m  =     25
n  =     10

 ** Output **

** k=   10 **
ierr =      0
Value of P.D.F. =    0.109
ierr =      0
Value of C.D.F. =    0.109

** k=   11 **
ierr =      0
Value of P.D.F. =    0.272
ierr =      0
Value of C.D.F. =    0.381

** k=   12 **
ierr =      0
Value of P.D.F. =    0.315
ierr =      0
Value of C.D.F. =    0.696

** k=   13 **
```

```
ierr =      0
Value of P.D.F. =     0.21
ierr =      0
Value of C.D.F. =    0.906

** k=   14 **
ierr =      0
Value of P.D.F. =   0.0803
ierr =      0
Value of C.D.F. =    0.986

** k=   15 **
ierr =      0
Value of P.D.F. =    0.014
ierr =      0
Value of C.D.F. =        1
```

Chapter 4

SAMPLE STATISTICS

4.1 INTRODUCTION

This library provides the following functions for calculating sample statistics.

- One Sample Basic Statistics
- Two Samples Basic Statistics
- Geometric Mean
- Moment
- m Samples Basic Statistics
- Harmonic Mean
- Root Mean Square
- Variance-Covariance Matrices
- Variance-Covariance Matrices (Grouped Data)
- Correlation Matrices
- Multiple Correlation Coefficients
- Partial Correlation Coefficients

4.1.1 Explanation

(1) **One Sample Basic Statistics**

The sum, mean, standard deviation, median of n observation data $\{x_i\}$ $(i = 1, \cdots, n)$ are defined by the following equations.

Sum :

$$t = \sum_{i=1}^{n} x_i$$

Mean :

$$\bar{x} = \frac{t}{n}$$

Standard deviation :

$$d = \sqrt{\frac{1}{\alpha}\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

Here, α is $n-1$ when an unbiased estimate is used or α is n when a sample variance is used.

Assume that we are given m classes of equal width $C_{i+1} = [c_i, c_i + h)$ $(i = 1, 2, \cdots, m)$ and the two classes $C_1 = (-\infty, c_1)$ and $C_{m+2} = [c_m + h, \infty)$, and that the following relationships are satisfied.

$$\begin{aligned} c_1 &= c_{min} \\ c_{i+1} &= c_i + h \quad i = 1, 2, \cdots, m \\ c_{max} &= c_m + h \end{aligned}$$

Here, h is the class width, which is defined as follows.

$$h = \frac{c_{max} - c_{min}}{m}$$

At this time, if the frequency of observation data in class C_i $(i = 1, 2, \cdots, m+2)$ is e_i, the frequency percentage f_i is defined as follows.

$$f_i = \frac{e_i}{n} \times 100 \quad i = 1, \cdots, m+2$$

Median :

$$p = a_{i-1} + \frac{h \times (N/2 - s_{i-1})}{g_i}$$

Here, a_{i-1} is lower boundary limit of class of median, g_i is frequency of class of median, s_{i-1} is frequency to class of median.

(2) **Two Samples Basic Statistics**

The sum, mean, and standard deviation of n two-sample observation data $\{x_i\}$ $(i = 1, \cdots, n)$ and $\{y_i\}$ $(i = 1, \cdots, n)$ are defined by the following equations.

Sum :

$$S_x = \sum_{i=1}^{n} x_i$$

$$S_y = \sum_{i=1}^{n} y_i$$

Mean :

$$\bar{x} = \frac{S_x}{n}$$

$$\bar{y} = \frac{S_y}{n}$$

Standard deviation :

$$SD_x = \sqrt{\frac{1}{\alpha}\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

Here, α is $n-1$ when an unbiased estimate is used or α is n when a sample variance is used.
Assume that for x_i, we are given m_x classes of equal width ${C_{i+1}}^{(x)} = [c_i, c_i + h_x)$ $(i = 1, 2, \cdots, m_x)$ and the two classes ${C_1}^{(x)} = (-\infty, c_1)$ and ${C_{m+2}}^{(x)} = [c_m + h_x, \infty)$. Assume that for y_i, we are given m_y classes of equal width ${C_{i+1}}^{(y)} = [d_i, d_i + h_y)$ $(i = 1, 2, \cdots, m_y)$ and the two classes ${C_1}^{(y)} = (-\infty, d_1)$ and ${C_{m+2}}^{(y)} = [d_m + h_y, \infty)$. Furthermore, assume that $(m_x + 2) \cdot (m_y + 2)$ classes on the $x - y$ plane defined as the direct product of x_i and y_i are defined by $C_{i,j} = (C_i^{(x)}, C_j^{(y)})$ $(i = 1, 2, \cdots, m_x + 2; j = 1, 2, \cdots, m_y + 2)$ and that the following relationships are satisfied.

$$\begin{aligned} c_1 &= c_{min} \\ c_{i+1} &= c_i + h_x \qquad i = 1, 2, \cdots, m_x \\ c_{max} &= c_{m_x} + h_x \end{aligned}$$

$$\begin{aligned} d_1 &= d_{min} \\ d_{i+1} &= d_i + h_y \qquad i = 1, 2, \cdots, m_y \\ d_{max} &= d_{m_y} + h_y \end{aligned}$$

Here, h_x and h_y are the class widths, which are defined as follows.

$$h_x = \frac{c_{max} - c_{min}}{m_x}$$

$$h_y = \frac{d_{may} - d_{min}}{m_y}$$

At this time, if the frequency of observation data in class $C_{i,j}$ is $e_{i,j}$, the frequency percentage $\{f_{ij}\}$ is defined as follows.

$$f_{ij} = \frac{e_{ij}}{n} \times 100 \quad i = 1, \cdots, m_y + 2, \quad j = 1, \cdots, m_x + 2$$

(3) **Geometric Mean**

Given a sample consisting of n observed values $\{x_i\}(i = 1, \cdots, n)$, the geometric mean and its standard deviation are defined by the following equations.
Geometric mean :

$$\mathrm{GM} = \left(\prod_{i=1}^{n} x_i\right)^{\frac{1}{n}}$$

Standard deviation :

$$\mathrm{GSD} \quad = \quad \exp\left(\sqrt{\frac{\sum_{i=1}^{n}(\log x_i)^2 - n(\log \mathrm{GM})^2}{\alpha}}\right)$$

Here, α is $n-1$ when an unbiased estimate is used or α is n when a sample variance is used.

(4) **Moment**

Assume that we are given m classes of equal width $C_i = [x_i, x_i + h)$ $(i = 1, 2, \cdots, m)$, and that the following relationships are satisfied.

$$\begin{aligned} x_1 &= x_{min} \\ x_{i+1} &= x_i + h \quad i = 1, 2, \cdots, m \\ x_{max} &= x_m + h \end{aligned}$$

Here, h is the class width, which is defined as follows.

$$h = \frac{x_{max} - x_{min}}{m}$$

At this time, if the frequency of observation data in class C_i is f_i, the linear moment about the origin and the moment of order r about the mean value are defined as follows. Linear moment about the origin :

$$\mu'_1 = \frac{\sum_{i=1}^{m} f_i \hat{x}_i}{\sum_{i=1}^{m} f_i}$$

Moment of order r about the mean value :

$$\mu_r = \frac{\sum_{i=1}^{m} [f_i (\hat{x}_i - \mu'_1)^r]}{\sum_{i=1}^{m} f_i} \qquad (r = 2, 3, \cdots)$$

$\hat{x}_i$, which represents the class value of each class, is defined as follows.

$$\hat{x}_i = x_{min} + (i - 0.5)h$$

(5) m **Samples Basic Statistics**

Given m samples consisting of n observed values $\{x_{ij}\}$, $(i = 1, \cdots, n;\ j = 1, \cdots, m)$, the basic statistics (sum, mean, sum of squares of deviation, variance, and standard deviation) for each sample are defined by the following equations.

Sum :

$$t_j = \sum_{i=1}^{n} x_{ij}, \qquad (j = 1, \cdots, m)$$

Mean :

$$\bar{x}_j = \frac{t_j}{n}, \quad (j = 1, \cdots, m)$$

Sum of squares of deviation :

$$s_j = \sum_{i=1}^{n} (x_{ij} - \bar{x}_j)^2, \quad (j = 1, \cdots, m)$$

Variance :

$$v_j = \frac{s_j}{\alpha}, \quad (j = 1, \cdots, m)$$

Standard deviation :

$$d_j = \sqrt{v_j}, \quad (j = 1, \cdots, m)$$

Here, α is $n - 1$ when an unbiased estimate is used or α is n when a sample variance is used.

(6) **Harmonic Mean**

Given a sample consisting of n observed values $\{x_i\}(i = 1, \cdots, n)$, the harmonic mean is defined by the following equations.

Harmonic mean :

$$\mathrm{HM} = \frac{1}{\frac{1}{n}\sum_{i=1}^{n} \frac{1}{x_i}}$$

(7) **Root Mean Square**

Given a sample consisting of n observed values $\{x_i\}(i = 1, \cdots, n)$, the root mean square is defined by the following equations.

Root mean square :

$$\mathrm{SM} = \sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}$$

(8) **Variance-Covariance Matrices**

Given m samples consisting of n observed values $\{x_{ki}\}(k = 1, \cdots, n; i = 1, \cdots, m)$, the variance-covariance among the samples $d_{i,j}$ is defined as follows.

$$d_{ij} = \frac{s_{ij}}{\alpha} \qquad i, j = 1, \cdots, m$$

Here, α is n when a sample covariance is used or α is $n - 1$ when an unbiased covariance is used. s_{ij} is defined as follows.

$$s_{ij} = \sum_{k=1}^{n} (x_{ki} - \bar{x}_i)(x_{kj} - \bar{x}_j), \qquad i = 1, \cdots, m; j = 1, \cdots, m$$

$\bar{x}_j$ is defined as follows.

$$\bar{x}_j = \frac{\sum_{i=1}^{n} x_{ij}}{n}, \qquad j = 1, \cdots, m$$

(9) **Variance-Covariance Matrices (Grouped Data)**
Assume there are g groups, and for each group, we are given m samples consisting of n_r observed values $\{x_{ij}^{(r)}\}(i = 1, \cdots, n_r; j = 1, \cdots, m; r = 1, \cdots, g)$. The variance-covariance over all groups $d_{i,j}$ is defined as follows.

$$d_{ij} = \frac{\sum_{r=1}^{g} s_{ij}^{(r)}}{\sum_{r=1}^{g} \alpha_r} \qquad i, j = 1, \cdots, m$$

For each group, $s_{ij}^{(r)}$ (deviation sum of product matrix) is defined as follows.

$$s_{ij}^{(r)} = \sum_{k=1}^{n_r} (x_{ki}^{(r)} - \bar{x}_i^{(r)})(x_{kj}^{(r)} - \bar{x}_j^{(r)}) \qquad i, j = 1, \cdots, m$$

Here, α_r is n_r when a sample covariance is used or α_r is $n_r - 1$ when an unbiased covariance is used. Also, $\bar{x}_i^{(r)}$ is the mean of each group.

(10) **Correlation Matrices**
Given m samples consisting of n observed values $\boldsymbol{x}_i = \{x_{ki}\}(k = 1, \cdots, n; i = 1, \cdots, m)$, the correlation coefficient r_{ij} between samples $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ is defined as follows.

$$r_{ij} = \frac{s_{ij}}{\sqrt{s_{ii} \cdot s_{jj}}}, \qquad i = 1, \cdots, m; j = 1, \cdots, m$$

s_{ij} is defined as follows.

$$s_{ij} = \sum_{k=1}^{n} (x_{ki} - \bar{x}_i)(x_{kj} - \bar{x}_j), \qquad i = 1, \cdots, m; j = 1, \cdots, m$$

$\bar{x}_j$ is defined as follows.

$$\bar{x}_j = \frac{\sum_{i=1}^{n} x_{ij}}{n}, \qquad j = 1, \cdots, m$$

Also, the matrix $R = (r_{ij})$ is called the correlation coefficient matrix. The correlation coefficients have the following properties.

- $|r_{ij}| \leq 1$
- $r_{ij} = r_{ji}$

(11) **Multiple Correlation Coefficients** and **Partial Correlation Coefficients**
For m variates X_i $(i = 1, \cdots, m)$, let $\bar{x_i}$ be the mean of each variate, ${\sigma_i}^2$ be the variance, and r_{ij} be the correlation coefficient of variates X_i and X_j. The multiple correlation coefficient $r_{p \cdot 1, \cdots, n}$, which is defined as the maximum value of the correlation coefficients of X_p $(l \geq p \geq m)$ and the linear expression $U = b + \sum_{i=1}^{l} c_i X_i$ of $X_1, \cdots, X_l$, can be calculated as follows.

$$r_{p \cdot 1, \cdots, l} = \sqrt{1 - \frac{\Delta}{\Delta_{pp}}}$$

Here, Δ and Δ_{ij} are the determinant and cofactor matrix of the matrix having the correlation coefficients r_{ij} $(i, j = 1, \cdots, l, p)$ as elements. Representing the value of U corresponding to this maximum value by $\hat{X}_p$, this can be expressed as follows.

$$\hat{X}_p = \bar{x}_p - \sum_{i=1}^{l} \sigma_p \frac{\Delta_{ip}}{\Delta_{pp}} \frac{X_i - \bar{x}_i}{\sigma_i}$$

$X_p - \hat{X}_p$, which is considered to be the variate when the influence of $X_1, \cdots, X_l$ has been eliminated from X_p, is called the remainder. The partial correlation coefficient $r_{p,q\cdot 1,\cdots,l}$ $(l \geq p, q \geq m)$, which is defined as the correlation coefficient of $X_p - \hat{X}_p$ and $X_q - \hat{X}_q$, can be calculated as follows.

$$r_{p,q\cdot 1,\cdots,l} = -\frac{\Delta_{pq}}{\sqrt{\Delta_{pp}\Delta_{qq}}}$$

4.1.2 Reference Bibliography

(1) Spiegel, M. R. , "Theory and Problems of Probability and Statistics", Schaum's outline series, McGraw-Hill, Inc. (1975).

4.2 BASIC STATISTICS

4.2.1 ASL_d2ba1t, ASL_r2ba1t
One Sample Basic Statistics

(1) **Function**

The ASL_d2ba1t or ASL_r2ba1t obtains the minimum, maximum, sum, mean, standard deviation, median, frequency, and frequency percentage of one-sample observation data. It also obtains the minimum, maximum, sum, mean, standard deviation, median, frequency, and frequency percentage when observation data is added.

The sum, mean, standard deviation and median of n observation data $\{x_i\}$ $(i = 1, \cdots, n)$ are defined by the following equations.

Sum :

$$t = \sum_{i=1}^{n} x_i$$

Mean :

$$\bar{x} = \frac{t}{n}$$

Standard deviation :

$$d = \sqrt{\frac{1}{\alpha}\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

Here, α is $n-1$ when an unbiased estimate is used or α is n when a sample variance is used.

Also, consider m classes of equal width $C_{i+1} = [c_i, c_i + h)$ $(i = 1, 2, \cdots, m)$ and the two classes $C_1 = (-\infty, c_1)$ and $C_{m+2} = [c_m + h, \infty)$, and assume that the following relationships are satisfied.

$$\begin{aligned} c_1 &= c_{min} \\ c_{i+1} &= c_i + h \quad i = 1, 2, \cdots, m \\ c_{max} &= c_m + h \end{aligned}$$

The ASL_d2ba1t or ASL_r2ba1t obtains the frequency e_i of the observation data in each class and the frequency percentage f_i, which is defined as follows.

$$f_i = \frac{e_i}{n} \times 100 \quad i = 1, \cdots, m+2$$

h is the class width, which is defined as follows.

$$h = \frac{c_{max} - c_{min}}{m}$$

Median :

$$p = a_{i-1} + \frac{h \times (N/2 - s_{i-1})}{g_i}$$

Here, a_{i-1} is the lower bound of the class of the median, g_i is the frequency of the class of the median, and s_{i-1} is the frequency to the class of the median.

(2) **Usage**

Double precision:

ierr = ASL_d2ba1t (a, n, nc, bl, bu, &ns, stat, ifrq1, freq2, isw, iwk, wk);

Single precision:

ierr = ASL_r2ba1t (a, n, nc, bl, bu, &ns, stat, ifrq1, freq2, isw, iwk, wk);

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	a	{D∗ / R∗}	n	Input	Sequence of observed values $\{x_i\}$
2	n	I	1	Input	Number of observed values n
3	nc	I	1	Input	Number of classes $m+2$
4	bl	{D / R}	1	Input	Value of c_{min}
5	bu	{D / R}	1	Input	Value of c_{max}
6	ns	I*	1	Input	Number of observed values before adding observed values (for isw=0 or 2, no initial setting is required)
				Output	Number of observed values n
7	stat	{D∗ / R∗}	6	Output	Basic statistics of observed values (See Note (b))
8	ifrq1	I*	nc	Output	Frequency for each class $\{e_i\}$
9	freq2	{D∗ / R∗}	nc	Output	Frequency percentage for each class $\{f_i\}$
10	isw	I	1	Input	Processing switch 0: Perform calculations using an unbiased estimate (no previously established basic statistics) 1: Perform calculations using an unbiased estimate (added observed values) 2: Perform calculations using a sample variance (no previously established basic statistics) 3: Perform calculations using a sample variance (added observed values)

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
11	iwk	I*	nc	Work	Work area
12	wk	$\left\{\begin{matrix}\text{D}* \\ \text{R}*\end{matrix}\right\}$	nc + n	Work	Work area
13	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) isw = 0, 1, 2, 3

(b) n $\geq$ 1

(c) nc $\geq$ 3

(d) bu $>$ bl

(e) ns $\geq$ 1 (When isw=1 or 3)

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
1000	Restriction (a) was not satisfied.	Processing continues with isw=0.
1010	An attempt was made to obtain the standard deviation when isw=0 and n=1.	The absolute value maximum that can be represented is set for the standard deviation.
1020	bu<bl occurred.	bu and bl are switched, and processing continues.
3000	Restriction (b), (c) was not satisfied.	Processing is aborted.
3010	bu=bl occurred.	
3020	Restriction (e) was not satisfied.	

(6) **Notes**

(a) When an observed value $<$ bl, it is included in the frequency for ifrq1[0].
When an observed value $\geq$ bu, it is included in the frequency for ifrq1[nc $-$ 1].

(b) The basic statistics are stored as follows in the array stat.
stat[0]: Minimum of observed values
stat[1]: Maximum of observed values
stat[2]: Sum of observed values t
stat[3]: Mean of observed values $\bar{x}$
stat[4]: Standard deviation of observed values d
stat[5]: Median of observed values p

(c) To obtain the basic statistics when observed values are added, use the contents of nc, bl, bu, ns, stat, ifrq1, freq2 and wk which were calculated before adding the observed values, set the added observed values for a and the number of added observed values for n, set isw to 1 or 3, and perform the calculation. However, when obtaining the standard deviation, you must set the isw value so that the calculation is performed using a sample variance following a calculation that used a sample variance the previous time or so that the calculation is performed using an unbiased estimate following a calculation that used an unbiased estimate the previous time.

(d) When there are an extremely large number of data values that are widely dispersed, better results are obtained by grouping them into data having absolute values of the same relative size and adding them to the samples in increasing order of size.

(e) Statistics obtained when calculations are performed using an unbiased estimate can be applied to a population for which sampling with replacement is performed from an infinite or finite population. Statistics obtained when calculations are performed using a sample variance can be applied to a population for which the population and sample match.

(7) **Example**

(a) Problem

Obtain the minimum, maximum, sum, mean, standard deviation, frequencies, and frequency percentages of the following one-sample observation data.

$$\{x_i\} = \{-10, -9, -8, -6, -5, -4, -3, -2, -1, 0, 1\}$$

Also, obtain the minimum, maximum, sum, mean, standard deviation, frequencies, and frequency percentages when the following one-sample observation data is added.

$$\{y_i\} = \{2, 3, 4, 5, 6, 7, 8, 9, 10, 11\}$$

(b) Input data

First processing :

One-sample observation data $\{x_i\}$, $\mathrm{n} = 11$, $\mathrm{nc} = 7$, $\mathrm{bl} = -6.5$, $\mathrm{bu} = 5.8$ and $\mathrm{isw} = 0$.

Second processing :

One-sample observation data $\{y_i\}$, $\mathrm{n} = 10$, $\mathrm{nc} = 7$, $\mathrm{bl} = -6.5$, $\mathrm{bu} = 5.8$ and $\mathrm{isw} = 1$.

(c) Main program

```
/*      C interface example for ASL_d2ba1t */

#include <stdio.h>
#include <stdlib.h>
#include <asl.h>

int main()
{
    double *a,*b;
    int n;
    int nc;
    double bl;
    double bu;
    int ns;
    double stat[6];
    int *ifrq1;
    double *freq2;
    int isw;
    int *iwk;
    double *wk;
    int ierr;
    int i,j;
    FILE *fp;

    fp = fopen( "d2ba1t.dat", "r" );
    if( fp == NULL )
    {
        printf( "file open error\n" );
        return -1;
    }

    printf( "    *** ASL_d2ba1t ***\n" );
    printf( "\n    ** First processing **\n" );
    printf( "\n     ** Input **\n\n" );

    nc = 7;
    isw =0;
    fscanf( fp, "%d", &n );
    fscanf( fp, "%lf", &bl );
    fscanf( fp, "%lf", &bu );
```

```
a = ( double * )malloc((size_t)( sizeof(double) * n ));
if( a == NULL )
{
    printf( "no enough memory for array a\n" );
    return -1;
}

ifrq1 = ( int * )malloc((size_t)( sizeof(int) * nc ));
if( ifrq1 == NULL )
{
    printf( "no enough memory for array ifrq1\n" );
    return -1;
}

freq2 = ( double * )malloc((size_t)( sizeof(double) * nc ));
if( freq2 == NULL )
{
    printf( "no enough memory for array freq2\n" );
    return -1;
}

iwk = ( int * )malloc((size_t)( sizeof(int) * nc ));
if( ifrq1 == NULL )
{
    printf( "no enough memory for array ifrq1\n" );
    return -1;
}

wk = ( double * )malloc((size_t)( sizeof(double) * (nc+n) ));
if( wk == NULL )
{
    printf( "no enough memory for array wk\n" );
    return -1;
}

printf( "\n\tn = %6d nc = %6d\n", n, nc );
printf( "\n\tbl = %8.3g bu = %8.3g\n", bl, bu );
printf( "\n\tisw = %6d\n", isw );

printf( "\n\tObservations\n" );
for( i=0 ; i<n ; i++ )
{
    fscanf( fp, "%lf", &a[i] );
    if( i%5 == 0 )
    {
        printf( "\n\t" );
    }
    printf( "%8.3g", a[i] );
}
printf( "\n" );

ierr = ASL_d2ba1t(a, n, nc, bl, bu, &ns, stat, ifrq1, freq2, isw, iwk, wk);

printf( "\n     ** Output **\n\n" );
printf( "\tierr = %6d\n", ierr );
printf( "\n\tns   = %6d\n\n", ns );

printf("\t    Minimum   Maximum      Sum       Mean    Standard     Median \n" );
printf("\t                                             deviation            \n" );
printf("\t-------------------------------------------------------------------\n" );
printf("\t");
for( j=0 ; j<6 ; j++ )
{
    printf( "%8.3g   ", stat[j] );
}
printf( "\n" );

printf( "\n\tFrequencies \tPercent frequencies\n\n" );
for( i=0 ; i<nc ; i++ )
{
    printf( "\t%6d     \t%8.3g\n", ifrq1[i],freq2[i] );
}

printf( "\n\n    ** Continuous processing **\n" );
printf( "\n     ** Input **\n\n" );

isw =1;
fscanf( fp, "%d", &n );

b = ( double * )malloc((size_t)( sizeof(double) * n ));
if( b == NULL )
{
    printf( "no enough memory for array b\n" );
    return -1;
}

printf( "\n\tn = %6d nc = %6d\n", n, nc );
printf( "\n\tbl = %8.3g bu = %8.3g\n", bl, bu );
printf( "\n\tns  = %6d\n", ns );
```

```
    printf( "\n\tisw = %6d\n", isw );

    printf( "\n\tObservations\n" );
    for( i=0 ; i<n ; i++ )
    {
        fscanf( fp, "%lf", &b[i] );
        if( i%5 == 0 )
        {
            printf( "\n\t" );
        }
        printf( "%8.3g", b[i] );
    }
    printf( "\n" );

    fclose( fp );

    ierr = ASL_d2ba1t(b, n, nc, bl, bu, &ns, stat, ifrq1, freq2, isw, iwk, wk);

    printf( "\n     ** Output **\n\n" );
    printf( "\tierr = %6d\n", ierr );
    printf( "\n\tns   = %6d\n\n", ns );

    printf("\t    Minimum   Maximum      Sum       Mean    Standard     Median \n" );
    printf("\t                                             deviation            \n" );
    printf("\t-------------------------------------------------------------------\n" );
    printf("\t");
    for( j=0 ; j<6 ; j++ )
    {
        printf( "%8.3g   ", stat[j] );
    }
    printf( "\n" );

    printf( "\n\tFrequencies \tPercent frequencies\n\n" );
    for( i=0 ; i<nc ; i++ )
    {
        printf( "\t%6d     \t%8.3g\n", ifrq1[i],freq2[i] );
    }

    free( a );
    free( ifrq1 );
    free( freq2 );
    free( iwk );
    free( wk );
    free( b );

    return 0;
}
```

(d) Output results

```
*** ASL_d2ba1t ***

** First processing **

 ** Input **


n =     11 nc =      7

bl =     -6.5 bu =      5.8

isw =      0

Observations

     -10      -9      -8      -6      -5
      -4      -3      -2      -1       0
       1

 ** Output **

ierr =      0

ns   =     11

    Minimum   Maximum      Sum       Mean    Standard     Median
                                           deviation
-------------------------------------------------------------------
     -10          1        -47      -4.27       3.69      -3.63

Frequencies     Percent frequencies

     3          27.3
     2          18.2
     3          27.3
     2          18.2
     1          9.09
     0             0
     0             0
```

```
** Continuous processing **
 ** Input **

n =     10 nc =      7

bl =     -6.5 bu =      5.8

ns  =     11

isw =      1

Observations

       2       3       4       5       6
       7       8       9      10      11

 ** Output **

ierr =      0

ns   =     21
```

Minimum	Maximum	Sum	Mean	Standard deviation	Median
-10	11	18	0.857	6.43	1.29

Frequencies	Percent frequencies
3	14.3
2	9.52
3	14.3
2	9.52
3	14.3
2	9.52
6	28.6

4.2.2 ASL_d2ba2s, ASL_r2ba2s Two Samples Basic Statistics

(1) **Function**

The ASL_d2ba2s or ASL_r2ba2s obtains the minimum, maximum, sum, mean, standard deviation, frequency, and frequency percentage of two-sample observation data. It also obtains the minimum, maximum, sum, mean, standard deviation, frequency, and frequency percentage when two-sample observation data is added. The sum, mean, and standard deviation of n two-sample observation data $\{x_i\}$ $(i = 1, \cdots, n)$ and $\{y_i\}$ $(i = 1, \cdots, n)$ are defined by the following equations.

Sum :

$$S_x = \sum_{i=1}^{n} x_i$$

$$S_y = \sum_{i=1}^{n} y_i$$

Mean :

$$\bar{x} = \frac{S_x}{n}$$

$$\bar{y} = \frac{S_y}{n}$$

Standard deviation :

$$SD_x = \sqrt{\frac{1}{\alpha}\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

$$SD_y = \sqrt{\frac{1}{\alpha}\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

Here, α is $n-1$ when an unbiased estimate is used or α is n when a sample variance is used.

Also, for x_i, consider m_x classes of equal width ${C_{i+1}}^{(x)} = [c_i, c_i + h_x)$ $(i = 1, 2, \cdots, m_x)$ and the two classes ${C_1}^{(x)} = (-\infty, c_1)$ and ${C_{m+2}}^{(x)} = [c_m + h_x, \infty)$, and for y_i, consider m_y classes of equal width ${C_{i+1}}^{(y)} = [d_i, d_i + h_y)$ $(i = 1, 2, \cdots, m_y)$ and the two classes ${C_1}^{(y)} = (-\infty, d_1)$ and ${C_{m+2}}^{(y)} = [d_m + h_y, \infty)$, and assume that the following relationships are satisfied.

$$\begin{aligned} c_1 &= c_{min} \\ c_{i+1} &= c_i + h_x \qquad i = 1, 2, \cdots, m_x \\ c_{max} &= c_{m_x} + h_x \end{aligned}$$

$$\begin{aligned} d_1 &= d_{min} \\ d_{i+1} &= d_i + h_y \qquad i = 1, 2, \cdots, m_y \\ d_{max} &= d_{m_y} + h_y \end{aligned}$$

On the $x - y$ plane defined as the direct product of x_i and y_i, define $(m_x + 2) \cdot (m_y + 2)$ classes as $C_{i,j} = (C_i^{(x)}, C_j^{(y)})$ $(i = 1, 2, \cdots, m_x + 2; j = 1, 2, \cdots, m_y + 2)$. The ASL_d2ba2s or ASL_r2ba2s obtains the

frequency $\{e_{ij}\}$ of observation data in each class and the frequency percentage $\{f_{ij}\}$, which is defined as follows.

$$f_{ij} = \frac{e_{ij}}{n} \times 100 \quad i = 1, \cdots, m_x + 2, \quad j = 1, \cdots, m_y + 2$$

h_x and h_y are the class widths, which are defined as follows.

$$h_x = \frac{c_{max} - c_{min}}{m_x}$$

$$h_y = \frac{d_{may} - d_{min}}{m_y}$$

(2) **Usage**

Double precision:

ierr = ASL_d2ba2s (x, n, y, ncx, ncy, blx, bux, bly, buy, &ns, stat, ifrq1, freq2, isw, wk);

Single precision:

ierr = ASL_r2ba2s (x, n, y, ncx, ncy, blx, bux, bly, buy, &ns, stat, ifrq1, freq2, isw, wk);

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	x	{D∗ / R∗}	n	Input	Sequence of observed values for sample X $\{x_i\}$
2	n	I	1	Input	Number of pairs of observed values n
3	y	{D∗ / R∗}	n	Input	Sequence of observed values for sample Y $\{y_i\}$
4	ncx	I	1	Input	Number of classes for sample X $m_x + 2$
5	ncy	I	1	Input	Number of classes for sample Y $m_y + 2$
6	blx	{D / R}	1	Input	Value of c_{min}
7	bux	{D / R}	1	Input	Value of c_{max}
8	bly	{D / R}	1	Input	Value of d_{min}
9	buy	{D / R}	1	Input	Value of d_{max}

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
10	ns	I*	1	Input	Number of pairs of observed values before adding observed values (for isw=0 or 2, no initial setting is required)
				Output	Number of pairs of observed values n
11	stat	$\left\{\begin{matrix}\text{D}* \\ \text{R}*\end{matrix}\right\}$	5×2	Output	Basic statistics of observed values (See Note (b))
12	ifrq1	I*	ncy×ncx	Output	Frequency distribution table $\{e_{ij}\}$
13	freq2	$\left\{\begin{matrix}\text{D}* \\ \text{R}*\end{matrix}\right\}$	ncy×ncx	Output	Frequency distribution table of percentages $\{f_{ij}\}$
14	isw	I	1	Input	Processing switch 0: Perform calculations using an unbiased estimate (no previously established basic statistics) 1: Perform calculations using an unbiased estimate (added observed values) 2: Perform calculations using a sample variance (no previously established basic statistics) 3: Perform calculations using a sample variance (added observed values)
15	wk	$\left\{\begin{matrix}\text{D}* \\ \text{R}*\end{matrix}\right\}$	See Contents	Work	Work area **Size**: ncx + ncy + 2 × n
16	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) isw = 0, 1, 2, 3

(b) n ≥ 1

(c) ncx ≥ 3

(d) ncy ≥ 3

(e) bux > blx

(f) buy > bly

(g) ns ≥ 1 (When isw=1 or 3)

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
1000	Restriction (a) was not satisfied.	Processing continues with isw=0.
1010	An attempt was made to obtain the standard deviation when isw=0 and n=1.	The absolute value maximum that can be represented is set for the standard deviation.
1020	bux<blx occurred.	bux and blx are switched, and processing continues.
1030	buy<bly occurred.	buy and bly are switched, and processing continues.
3000	Restriction (b), (c), (d) was not satisfied.	Processing is aborted.
3010	bux = blx or buy = bly occurred.	
3020	Restriction (g) was not satisfied.	

(6) **Notes**

(a) The frequencies of observed values $\{e_{ij}\}$ and frequency percentages $\{f_{ij}\}$ are stored in array ifrq1 and freq2, respectively as real matrix (two-dimensional array type) data (See Appendix A).

(b) The basic statistics are stored as follows in the array stat.

stat[0]: Minimum for sample X
stat[1]: Maximum for sample X
stat[2]: Sum for sample X S_x
stat[3]: Mean for sample X $\bar{x}$
stat[4]: Standard deviation for sample X SD_x
stat[5]: Minimum for sample Y
stat[6]: Maximum for sample Y
stat[7]: Sum for sample Y S_y
stat[8]: Mean for sample Y $\bar{y}$
stat[9]: Standard deviation for sample Y SD_y

(c) To obtain the basic statistics, frequencies, and frequency percentages when observed values are added, use the contents of ncx, ncy, blx, bux, bly, buy, ns, stat, ifrq1, freq2 and wk, which were calculated before adding the observed values, set the added pairs of observed values for x and y and the number of added pairs of observed values for n, set isw to 1 or 3, and perform the calculation. However, when obtaining the standard deviation, you must set the isw value so that the calculation is performed using a sample variance following a calculation that used a sample variance the previous time or so that the calculation is performed using an unbiased estimate following a calculation that used an unbiased estimate the previous time.

(d) When there are an extremely large number of data values that are widely dispersed, better results are obtained by grouping them into data having absolute values of the same relative size and adding them to the samples in increasing order of size.

(e) Statistics obtained when calculations are performed using an unbiased estimate can be applied to a population for which sampling with replacement is performed from an infinite or finite population. Statistics obtained when calculations are performed using a sample variance can be applied to a population for which the population and sample match.

(7) **Example**

(a) Problem

Obtain the minimum, maximum, sum, mean, standard deviation, frequencies, and frequency percentages of the following two-sample observation data.

$$\{x_i\} = \{1, 3, 5, 7, 9, 11, 13\}$$
$$\{y_i\} = \{2, 4, 6, 8, 10, 12, 14\}$$

Also, obtain the minimum, maximum, sum, mean, standard deviation, frequencies, and frequency percentages when the following two-sample observation data is added.

$$\{x'_i\} = \{15, 17, 19, 21\}$$
$$\{y'_i\} = \{16, 18, 20, 22\}$$

(b) Input data

First processing :

Two-sample observation data $\{x_i\},\ \{y_i\}$,

n $= 7,$ ncx $= 8,$ ncy $= 7,$ blx $= 5.5,$ bly $= 2.5,$

bux $= 11.5,$ buy $= 17.5$ and isw $= 0.$

Second processing :

Two-sample observation data $\{x'_i\},\ \{y'_i\}$,

n $= 4,$ ncx $= 8,$ ncy $= 7,$ blx $= 5.5,$ bly $= 2.5,$

bux $= 11.5,$ buy $= 17.5$ and isw $= 1.$

(c) Main program

```
/*      C interface example for ASL_d2ba2s */

#include <stdio.h>
#include <stdlib.h>
#include <asl.h>

int main()
{
    double *x,*x2;
    int n;
    double *y,*y2;
    int ncx,ncy;
    double blx,bux,bly,buy;
    int ns;
    double stat[5*2];
    int *ifrq1;
    double *freq2;
    int isw;
    double *wk;
    int ierr;
    int i,j;
    FILE *fp;

    fp = fopen( "d2ba2s.dat", "r" );
    if( fp == NULL )
    {
        printf( "file open error\n" );
        return -1;
    }

    printf( "    *** ASL_d2ba2s ***\n" );
    printf( "\n    ** First processing **\n" );
    printf( "\n     ** Input **\n\n" );

    fscanf( fp, "%d", &n );
    fscanf( fp, "%d", &ncx );
    fscanf( fp, "%d", &ncy );
    fscanf( fp, "%lf", &blx );
```

```
fscanf( fp, "%lf", &bux );
fscanf( fp, "%lf", &bly );
fscanf( fp, "%lf", &buy );
fscanf( fp, "%d", &isw );

x = ( double * )malloc((size_t)( sizeof(double) * n ));
if( x == NULL )
{
    printf( "no enough memory for array x\n" );
    return -1;
}

y = ( double * )malloc((size_t)( sizeof(double) * n ));
if( y == NULL )
{
    printf( "no enough memory for array y\n" );
    return -1;
}

ifrq1 = ( int * )malloc((size_t)( sizeof(int) * (ncy*ncx) ));
if( ifrq1 == NULL )
{
    printf( "no enough memory for array ifrq1\n" );
    return -1;
}

freq2 = ( double * )malloc((size_t)( sizeof(double) * (ncy*ncx) ));
if( freq2 == NULL )
{
    printf( "no enough memory for array freq2\n" );
    return -1;
}

wk = ( double * )malloc((size_t)( sizeof(double) * (ncx+ncy+2*n) ));
if( wk == NULL )
{
    printf( "no enough memory for array wk\n" );
    return -1;
}


printf( "\tn   = %6d\n", n );
printf( "\tncx = %6d\n", ncx );
printf( "\tblx = %8.3g\n", blx );
printf( "\tbux = %8.3g\n", bux );
printf( "\tncy = %6d\n", ncy );
printf( "\tbly = %8.3g\n", bly );
printf( "\tbuy = %8.3g\n", buy );
printf( "\tisw = %6d\n", isw );

for( i=0 ; i<n ; i++ )
{
    fscanf( fp, "%lf", &x[i] );
}
for( i=0 ; i<n ; i++ )
{
    fscanf( fp, "%lf", &y[i] );
}
printf("\n\tObservations x and y\n");
for( i=0 ; i<n ; i++ )
{
    printf( "\t%8.3g%8.3g\n", x[i],y[i] );
}

ierr = ASL_d2ba2s(x,n,y,ncx,ncy,blx,bux,bly,buy,
        &ns,stat,ifrq1,freq2,isw,wk);

printf( "\n     ** Output **\n\n" );
printf( "\tierr = %6d\n", ierr );
printf( "\tns   = %6d\n\n", ns );

printf("\t      Minimum   Maximum      Sum       Mean    Standard \n" );
printf("\t                                                deviation\n" );
printf("\t-------------------------------------------------------------\n" );
printf("\tx ");
for( j=0 ; j<5 ; j++ )
{
    printf( "%8.3g   ", stat[j] );
}
printf( "\n" );
printf("\ty ");
for( j=0 ; j<5 ; j++ )
{
    printf( "%8.3g   ", stat[5+j] );
}
printf( "\n" );

printf( "\n\tFrequencies\n\n" );
```

```
for( i=0 ; i<ncy ; i++ )
{
    printf( "\t" );
    for( j=0 ; j<ncx ; j++ )
    {
        printf( "%6d ", ifrq1[i+ncy*j] );
    }
    printf( "\n" );
}

printf( "\n\tPercent frequencies\n\n" );
for( i=0 ; i<ncy ; i++ )
{
    printf( "\t" );
    for( j=0 ; j<ncx ; j++ )
    {
        printf( "%7.3g ", freq2[i+ncy*j] );
    }
    printf( "\n" );
}

printf( "\n\n    ** Continuous processing **\n" );
printf( "\n     ** Input **\n\n" );

fscanf( fp, "%d", &n );
fscanf( fp, "%d", &isw );
x2 = ( double * )malloc((size_t)( sizeof(double) * n ));
if( x2 == NULL )
{
    printf( "no enough memory for array x2\n" );
    return -1;
}

y2 = ( double * )malloc((size_t)( sizeof(double) * n ));
if( y2 == NULL )
{
    printf( "no enough memory for array y2\n" );
    return -1;
}


printf( "\tn   = %6d\n", n );
printf( "\tncx = %6d\n", ncx );
printf( "\tblx = %8.3g\n", blx );
printf( "\tbux = %8.3g\n", bux );
printf( "\tncy = %6d\n", ncy );
printf( "\tbly = %8.3g\n", bly );
printf( "\tbuy = %8.3g\n", buy );
printf( "\tns = %6d\n", ns );
printf( "\tisw = %6d\n", isw );

for( i=0 ; i<n ; i++ )
{
    fscanf( fp, "%lf", &x2[i] );
}
for( i=0 ; i<n ; i++ )
{
    fscanf( fp, "%lf", &y2[i] );
}

printf("\n\tObservations x and y\n");
for( i=0 ; i<n ; i++ )
{
    printf( "\t%8.3g%8.3g\n", x2[i], y2[i] );
}

fclose( fp );

ierr = ASL_d2ba2s(x2, n, y2, ncx, ncy, blx, bux, bly, buy, &ns, stat, ifrq1, freq2, isw, wk);

printf( "\n     ** Output **\n\n" );
printf( "\tierr = %6d\n", ierr );
printf( "\tns   = %6d\n\n", ns );

printf("\t      Minimum   Maximum      Sum       Mean    Standard \n" );
printf("\t                                               deviation\n" );
printf("\t-------------------------------------------------------\n" );
printf("\tx ");
for( j=0 ; j<5 ; j++ )
{
    printf( "%8.3g   ", stat[j] );
}
printf( "\n" );
printf("\ty ");
for( j=0 ; j<5 ; j++ )
{
    printf( "%8.3g   ", stat[5+j] );
```

```
    }
    printf( "\n" );


    printf( "\n\tFrequencies\n" );
    for( i=0 ; i<ncy ; i++ )
    {
        printf( "\t" );
        for( j=0 ; j<ncx ; j++ )
        {
            printf( "%6d ", ifrq1[i+ncy*j] );
        }
        printf( "\n" );
    }

    printf( "\n\tPercent frequencies\n\n" );
    for( i=0 ; i<ncy ; i++ )
    {
        printf( "\t" );
        for( j=0 ; j<ncx ; j++ )
        {
            printf( "%7.3g ", freq2[i+ncy*j] );
        }
        printf( "\n" );
    }

    free( x );
    free( y );
    free( ifrq1 );
    free( freq2 );
    free( wk );
    free( x2 );
    free( y2 );

    return 0;
}
```

(d) Output results

```
*** ASL_d2ba2s ***

** First processing **

 ** Input **

n   =      7
ncx =      8
blx =      5.5
bux =     11.5
ncy =      7
bly =      2.5
buy =     17.5
isw =      0

Observations x and y
       1       2
       3       4
       5       6
       7       8
       9      10
      11      12
      13      14

 ** Output **

ierr =      0
ns   =      7

      Minimum   Maximum      Sum       Mean    Standard
                                             deviation
-------------------------------------------------------
x         1         13         49          7       4.32
y         2         14         56          8       4.32

Frequencies

     1      0      0      0      0      0      0      0
     1      0      0      0      0      0      0      0
     1      0      1      0      0      0      0      0
     0      0      0      0      1      0      0      0
     0      0      0      0      0      0      1      1
     0      0      0      0      0      0      0      0
     0      0      0      0      0      0      0      0

Percent frequencies

   14.3       0       0       0       0       0       0       0
   14.3       0       0       0       0       0       0       0
   14.3       0    14.3       0       0       0       0       0
      0       0       0       0    14.3       0       0       0
      0       0       0       0       0       0    14.3    14.3
```

```
      0       0       0       0       0       0       0       0
      0       0       0       0       0       0       0       0

** Continuous processing **

 ** Input **

n   =      4
ncx =      8
blx =      5.5
bux =     11.5
ncy =      7
bly =      2.5
buy =     17.5
ns =      7
isw =      1

Observations x and y
      15      16
      17      18
      19      20
      21      22

 ** Output **

ierr =      0
ns   =     11

      Minimum   Maximum      Sum       Mean    Standard
                                             deviation
-------------------------------------------------------------
x         1         21        121         11       6.63
y         2         22        132         12       6.63

Frequencies
     1      0      0      0      0      0      0      0
     1      0      0      0      0      0      0      0
     1      0      1      0      0      0      0      0
     0      0      0      0      1      0      0      0
     0      0      0      0      0      0      1      1
     0      0      0      0      0      0      0      1
     0      0      0      0      0      0      0      3

Percent frequencies

   9.09       0       0       0       0       0       0       0
   9.09       0       0       0       0       0       0       0
   9.09       0    9.09       0       0       0       0       0
      0       0       0       0    9.09       0       0       0
      0       0       0       0       0       0    9.09    9.09
      0       0       0       0       0       0       0    9.09
      0       0       0       0       0       0       0    27.3
```

4.2.3 ASL_d2bams, ASL_r2bams
m Samples Basic Statistics

(1) **Function**

Given m samples consisting of n observed values $\{x_{ij}\}$, $(i = 1, \cdots, n;\ j = 1, \cdots, m)$, the ASL_d2bams or ASL_r2bams obtains the basic statistics (sum, mean, sum of squares of deviation, variance, and standard deviation) for each sample. It also obtains the basic statistics when n observed values $\{y_{ij}\}$, $(i = 1, \cdots, n;\ j = 1, \cdots, m)$ are added to each of the m samples for which the basic statistics are known.

The basic statistics for m samples consisting of n observed values $\{x_{ij}\}$, $(i = 1, \cdots, n;\ j = 1, \cdots, m)$ are defined by the following equations.

Sum :

$$t_j = \sum_{i=1}^{n} x_{ij}, \quad (j = 1, \cdots, m)$$

Mean :

$$\bar{x}_j = \frac{t_j}{n}, \quad (j = 1, \cdots, m)$$

Sum of squares of deviation :

$$s_j = \sum_{i=1}^{n} (x_{ij} - \bar{x}_j)^2, \quad (j = 1, \cdots, m)$$

Variance :

$$v_j = \frac{s_j}{\alpha}, \quad (j = 1, \cdots, m)$$

Standard deviation :

$$d_j = \sqrt{v_j}, \quad (j = 1, \cdots, m)$$

Here, α is $n - 1$ when an unbiased estimate is used or α is n when a sample variance is used.

(2) **Usage**

Double precision:

```
ierr = ASL_d2bams  (a, na, n, m, &ns, stat, isw);
```

Single precision:

```
ierr = ASL_r2bams  (a, na, n, m, &ns, stat, isw);
```

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	a	{D∗ / R∗}	na×m	Input	Matrix in which observed values are stored (x_{ij}) or (y_{ij})
2	na	I	1	Input	Adjustable dimension of array a
3	n	I	1	Input	Number of observed values per sample stored in array a n
4	m	I	1	Input	Number of samples m
5	ns	I*	1	Input	Number of observed values per sample before adding observed values (for isw=0 or 2, no initial setting is required)
				Output	Number of observed values per sample n
6	stat	{D∗ / R∗}	m×5	Input	Basic statistics before adding observed values (See Note (a)) (for isw=0 or 2, no initial setting is required)
				Output	Obtained basic statistics (See Note (a))
7	isw	I	1	Input	Processing switch 0: Perform calculations using an unbiased estimate (no previously established basic statistics) 1: Perform calculations using an unbiased estimate (added observed values) 2: Perform calculations using a sample variance (no previously established basic statistics) 3: Perform calculations using a sample variance (added observed values)
8	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) isw=0, 1, 2, 3

(b) na≥n≥1

(c) m≥1

(d) ns≥1 (When isw=1 or 3)

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
1000	Restriction (a) was not satisfied.	Processing continues with isw=0.
1010	An attempt was made to obtain an unbiased estimate when n=1.	The absolute value maximum that can be represented is set for the variance and standard deviation.
3000	Restriction (b) or (c) was not satisfied.	Processing is aborted.
3010	Restriction (d) was not satisfied.	

(6) **Notes**

(a) The basic statistics are stored as follows in the array stat.

stat$[(j-1)]$: Sum t_j
stat$[(j-1)+m]$: Mean $\bar{x}_j$
stat$[(j-1)+2\times m]$: Sum of squares of deviation s_j , $(j=1,\cdots,\mathrm{m})$
stat$[(j-1)+3\times m]$: Variance v_j
stat$[(j-1)+4\times m]$: Standard deviation d_j

(b) To obtain the basic statistics when the same numbers of observed values are added for each sample, use the contents of stat and ns, which were calculated before adding the observed values, set the added observed values for a and the number of added observed values for n, set isw to 1 or 3, and perform the calculation. However, when obtaining the variance or standard deviation, you must set the isw value so that the calculation is performed using a sample variance following a calculation that used a sample variance the previous time or so that the calculation is performed using an unbiased estimate following a calculation that used an unbiased estimate the previous time.

(c) When there are an extremely large number of data values that are widely dispersed, better results are obtained by grouping them into data having absolute values of the same relative size and adding them to the samples in increasing order of size.

(d) Statistics obtained when calculations are performed using an unbiased estimate can be applied to a population for which sampling with replacement is performed from an infinite or finite population. Statistics obtained when calculations are performed using a sample variance can be applied to a population for which the population and sample match.

(7) **Example**

(a) Problem

Obtain the basic statistics (sum, mean, sum of squares of deviation, variance, and standard deviation) for each sample when the observed values are given by matrix X shown below.

$$X = \begin{bmatrix} 30 & 35 & 44 & 44 & 45 \\ 424 & 365 & 346 & 349 & 297 \\ 246 & 219 & 255 & 252 & 256 \end{bmatrix}$$

Also, obtain the basic statistics (sum, mean, sum of squares of deviation, variance, and standard deviation) for each sample when the observed values given by matrix Y shown below are added.

$$Y = \begin{bmatrix} 18 & 21 & 56 & 21 & 45 \\ 2 & 2 & 3 & 1 & 3 \end{bmatrix}$$

(b) Input data

First processing :

Matrix X in which observed values are stored, na $= 100$, n $= 3$, m $= 5$ and isw $= 0$.

Second processing :

Matrix Y in which observed values are stored, na $= 100$, n $= 2$, m $= 5$ and isw $= 1$.

(c) Main program

```
/*      C interface example for ASL_d2bams */

#include <stdio.h>
#include <stdlib.h>
#include <asl.h>

int main()
{
    double *a;
    int na;
    int n;
    int m;
    int ns;
    double *stat;
    int isw;
    int ierr;
    int i,j;
    FILE *fp;

    fp = fopen( "d2bams.dat", "r" );
    if( fp == NULL )
    {
        printf( "file open error\n" );
        return -1;
    }

    printf( "    *** ASL_d2bams ***\n" );
    printf( "\n    ** First Processing **\n" );
    printf( "\n     ** Input **\n\n" );

    fscanf( fp, "%d", &na );
    fscanf( fp, "%d", &n );
    fscanf( fp, "%d", &m );
    fscanf( fp, "%d", &isw );

    a = ( double * )malloc((size_t)( sizeof(double) * (na*m) ));
    if( a == NULL )
    {
        printf( "no enough memory for array a\n" );
        return -1;
    }

    stat = ( double * )malloc((size_t)( sizeof(double) * (m*5) ));
    if( stat == NULL )
    {
        printf( "no enough memory for array stat\n" );
        return -1;
    }

    printf( "\tna  = %6d\n", na );
    printf( "\tn   = %6d\n", n  );
    printf( "\tm   = %6d\n", m  );
    printf( "\tisw = %6d\n", isw );
    printf( "\n\tObservations\n\n", isw );
    for( i=0 ; i<n ; i++ )
    {
        printf("\t");
        for( j=0 ; j<m ; j++ )
        {
            fscanf( fp, "%lf", &a[i+na*j] );
            printf( "%8.3g ", a[i+na*j] );
        }
        printf( "\n" );
    }

    ierr = ASL_d2bams(a, na, n, m, &ns, stat, isw);

    printf( "\n     ** Output **\n\n" );
    printf( "\tierr = %6d\n", ierr );
    printf( "\tns   = %6d\n\n", ns );

    printf( "\t    Sum       Mean      Sum of    Variance    Standard \n" );
    printf( "\t                        Squares               deviation\n" );
    printf( "\t-------------------------------------------------------\n" );
    for( i=0 ; i<m ; i++ )
```

```
    {
        printf("\t");
        for( j=0 ; j<5 ; j++ )
        {
            printf( "%8.3g   ", stat[i+m*j] );
        }
        printf( "\n" );
    }

    printf( "\n\n    ** Continuous Processing **\n" );
    printf( "\n     ** Input **\n\n" );

    fscanf( fp, "%d", &n );
    fscanf( fp, "%d", &isw );

    printf( "\tna  = %6d\n", na  );
    printf( "\tn   = %6d\n", n   );
    printf( "\tm   = %6d\n", m   );
    printf( "\tisw = %6d\n", isw );
    printf( "\n\tObservations\n\n", isw );
    for( i=0 ; i<n ; i++ )
    {
        printf("\t");
        for( j=0 ; j<m ; j++ )
        {
            fscanf( fp, "%lf", &a[i+na*j] );
            printf( "%8.3g ", a[i+na*j] );
        }
        printf( "\n" );
    }

    ierr = ASL_d2bams(a, na, n, m, &ns, stat, isw);

    printf( "\n     ** Output **\n\n" );
    printf( "\tierr = %6d\n", ierr );
    printf( "\tns   = %6d\n\n", ns );

    printf( "\t    Sum       Mean      Sum of    Variance    Standard \n" );
    printf( "\t                        Squares               deviation\n" );
    printf( "\t-------------------------------------------------------\n" );
    for( i=0 ; i<m ; i++ )
    {
        printf("\t");
        for( j=0 ; j<5 ; j++ )
        {
            printf( "%8.3g   ", stat[i+m*j] );
        }
        printf( "\n" );
    }

    fclose( fp );
    free( a );
    free( stat );

    return 0;
}
```

(d) Output results

```
*** ASL_d2bams ***

** First Processing **

 ** Input **

na  =    100
n   =      3
m   =      5
isw =      0

Observations

      30       35       44       44       45
     424      365      346      349      297
     246      219      255      252      256

 ** Output **

ierr =      0
ns   =      3

    Sum       Mean      Sum of    Variance    Standard
                        Squares               deviation
-------------------------------------------------------
     700        233   7.79e+04   3.89e+04         197
     619        206   5.47e+04   2.73e+04         165
     645        215    4.8e+04    2.4e+04         155
     645        215   4.86e+04   2.43e+04         156
     598        199   3.66e+04   1.83e+04         135
```

```
** Continuous Processing **

 ** Input **

na  =    100
n   =      2
m   =      5
isw =      1

Observations

      18       21       56       21       45
       2        2        3        1        3

 ** Output **

ierr =      0
ns   =      5

    Sum       Mean      Sum of    Variance    Standard
                        Squares               deviation
-----------------------------------------------------------
     720        144   1.38e+05   3.45e+04         186
     642        128      1e+05   2.51e+04         158
     704        141   9.07e+04   2.27e+04         151
     667        133   9.87e+04   2.47e+04         157
     646        129   7.43e+04   1.86e+04         136
```

4.2.4 ASL_d2bagm, ASL_r2bagm Geometric Mean

(1) **Function**

Given a sample consisting of n observed values $\{x_i\}(i = 1, \cdots, n)$, The ASL_d2bagm or ASL_r2bagm obtains the geometric mean and its standard deviation. Also, the ASL_d2bagm or ASL_r2bagm obtains the geometric mean and its standard deviation when n observed values $\{y_i\}(i = 1, \cdots, n)$ are added.

For a sample consisting of n observed values $\{x_i\}(i = 1, \cdots, n)$, the geometric mean and its standard deviation are defined by the following equations.

Geometric mean :

$$\mathrm{GM} = \left(\prod_{i=1}^{n} x_i \right)^{\frac{1}{n}}$$

Standard deviation :

$$\mathrm{GSD} = \exp\left(\sqrt{\frac{\sum_{i=1}^{n} (\log x_i)^2 - n(\log \mathrm{GM})^2}{\alpha}} \right)$$

Here, α is $n - 1$ when an unbiased estimate is used or α is n when a sample variance is used.

(2) **Usage**

Double precision:

```
ierr = ASL_d2bagm  (a, n, &ns, &gm, &gsd, isw);
```

Single precision:

```
ierr = ASL_r2bagm  (a, n, &ns, &gm, &gsd, isw);
```

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	a	{D∗ / R∗}	n	Input	Sequence of observed values $\{x_i\}$ or $\{y_i\}$
2	n	I	1	Input	Number of observed values n
3	ns	I*	1	Input	Number of observed values before adding observed values (for isw=0 or 2, no initial setting is required)
				Output	Number of observed values n
4	gm	{D∗ / R∗}	1	Input	Geometric mean before adding observed values (See Note (a)) (for isw=0 or 2, no initial setting is required)
				Output	Geometric mean that was obtained (See Note (a))
5	gsd	{D∗ / R∗}	1	Input	Standard deviation before adding observed values (See Note (a)) (for isw=0 or 2, no initial setting is required)
				Output	Standard deviation that was obtained (See Note (a))
6	isw	I	1	Input	Processing switch 0: Perform calculations using an unbiased estimate (no previously established basic statistics) 1: Perform calculations using an unbiased estimate (added observed values) 2: Perform calculations using a sample variance (no previously established basic statistics) 3: Perform calculations using a sample variance (added observed values)
7	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) isw = 0, 1, 2, 3

(b) n $\geq$ 1

(c) ns $\geq$ 1 (When isw=1 or 3)

(d) a[i − 1] > 0.0(i = 1, 2, $\cdots$, n)

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
1000	Restriction (a) was not satisfied.	Processing continues with isw=0.
1010	An attempt was made to obtain an unbiased estimate when n = 1.	The absolute value maximum that can be represented is set for the standard deviation.
3000	Restriction (b) was not satisfied.	Processing is aborted.
3010	Restriction (c) was not satisfied.	
3020	Restriction (d) was not satisfied.	

(6) **Notes**

(a) To obtain the statistics when observed values are added, use the contents of gm, gsd and ns, which were calculated before adding the observed values, set the added observed values for a and the number of added observed values for n, set isw to 1 or 3, and perform the calculation.

(b) When there are an extremely large number of data values that are widely dispersed, better results are obtained by grouping them into data having absolute values of the same relative size and adding them to the samples in increasing order of size.

(c) Statistics obtained when calculations are performed using an unbiased estimate can be applied to a population for which sampling with replacement is performed from an infinite or finite population. Statistics obtained when calculations are performed using a sample variance can be applied to a population for which the population and sample match.

(7) **Example**

(a) Problem
Given the following sequence of observed values, obtain the geometric mean and its standard deviation.

$$\{x_i\} = \{1100, 2630, 695, 3630, 1550, 1010, 2110, 736, 1260, 1690, 2680, 2520, 2030, 1280, 2400\}$$

Also, obtain the geometric mean and its standard deviation when the following sequence of observed values are added.

$$\{y_i\} = \{938, 1860, 2410, 3370, 1380, 2200, 2290, 1220, 1150\}$$

(b) Input data
First processing :

Sequence of observed values $\{x_i\}$, n = 15 and isw = 0.

Second processing :

Sequence of observed values $\{y_i\}$, n = 9 and isw = 1.

(c) Main program

```
/*      C interface example for ASL_d2bagm */

#include <stdio.h>
#include <stdlib.h>
#include <asl.h>
```

```
int main()
{
    double *a;
    int n;
    int ns;
    double gm;
    double gsd;
    int isw;
    int ierr;
    int i;
    FILE *fp;

    fp = fopen( "d2bagm.dat", "r" );
    if( fp == NULL )
    {
        printf( "file open error\n" );
        return -1;
    }

    printf( "    *** ASL_d2bagm ***\n" );
    printf( "\n    ** First processing **\n" );
    printf( "\n     ** Input **\n\n" );

    fscanf( fp, "%d", &n );
    fscanf( fp, "%d", &isw );

    a = ( double * )malloc((size_t)( sizeof(double) * n ));
    if( a == NULL )
    {
        printf( "no enough memory for array a\n" );
        return -1;
    }


    printf( "\tn   = %6d\n", n );
    printf( "\tisw = %6d\n", isw );
    printf( "\n\tObservations\n");
    for( i=0 ; i<n ; i++ )
    {
        if( i%5 == 0 )
        {
            printf( "\n\t" );
        }
        fscanf( fp, "%lf", &a[i] );
        printf( "%9.5g", a[i] );
    }

    ierr = ASL_d2bagm(a, n, &ns, &gm, &gsd, isw);

    printf( "\n\n     ** Output **\n\n" );
    printf( "\tierr = %9d\n", ierr );
    printf( "\tns   = %9d\n", ns );
    printf( "\tgm   = %9.5g\n", gm );
    printf( "\tgsd  = %9.3g\n", gsd );

    printf( "\n\n    ** Continuous processing **\n" );
    printf( "\n     ** Input **\n\n" );

    fscanf( fp, "%d", &n );
    fscanf( fp, "%d", &isw );

    printf( "\tn   = %6d\n\tns  = %6d\n", n, ns );
    printf( "\tisw = %6d\n", isw );
    printf( "\n\tObservations\n");
    for( i=0 ; i<n ; i++ )
    {
        if( i%5 == 0 )
        {
            printf( "\n\t" );
        }
        fscanf( fp, "%lf", &a[i] );
        printf( "%9.5g", a[i] );
    }

    fclose( fp );

    ierr = ASL_d2bagm(a, n, &ns, &gm, &gsd, isw);

    printf( "\n\n     ** Output **\n\n" );
    printf( "\tierr = %9d\n", ierr );
    printf( "\tns   = %9d\n", ns );
    printf( "\tgm   = %9.5g\n", gm );
    printf( "\tgsd  = %9.3g\n", gsd );

    free( a );

    return 0;
}
```

(d) Output results

```
*** ASL_d2bagm ***

** First processing **

 ** Input **

n   =     15
isw =      0

Observations

     1100     2630      695     3630     1550
     1010     2110      736     1260     1690
     2680     2520     2030     1280     2400

 ** Output **

ierr =         0
ns   =        15
gm   =    1634.7
gsd  =      1.64


** Continuous processing **

 ** Input **

n   =      9
ns  =     15
isw =      1

Observations

      938     1850     2410     3370     1380
     2200     2290     1220     1150

 ** Output **

ierr =         0
ns   =        24
gm   =    1669.2
gsd  =      1.58
```

4.2.5 ASL_d2bamo, ASL_r2bamo Moment

(1) **Function**

Assume that we are given m classes of equal width $C_i = [x_i, x_i + h)$ $(i = 1, 2, \cdots, m)$, and that the following relationships are satisfied.

$$\begin{aligned} x_1 &= x_{min} \\ x_{i+1} &= x_i + h \qquad i = 1, 2, \cdots, m \\ x_{max} &= x_m + h \end{aligned}$$

Here, h is the class width, which is defined as follows.

$$h = \frac{x_{max} - x_{min}}{m}$$

The ASL_d2bamo or ASL_r2bamo obtains the linear moment about the origin and the moment of order r about the mean value, which are defined as follows, where the frequency of observation data in class C_i is f_i.

Linear moment about the origin :

$$\mu'_1 = \frac{\sum_{i=1}^{m} f_i \hat{x}_i}{\sum_{i=1}^{m} f_i}$$

Moment of order r about the mean value :

$$\mu_r = \frac{\sum_{i=1}^{m} [f_i (\hat{x}_i - \mu'_1)^r]}{\sum_{i=1}^{m} f_i} \qquad (r = 2, 3, \cdots)$$

$\hat{x}_i$, which represents the class value of each class, is defined as follows.

$$\hat{x}_i = x_{min} + (i - 0.5)h$$

(2) **Usage**

Double precision:

```
ierr = ASL_d2bamo  (f, m, xmax, xmin, np, &xm, wk);
```

Single precision:

```
ierr = ASL_r2bamo  (f, m, xmax, xmin, np, &xm, wk);
```

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	f	{D∗ / R∗}	m	Input	Frequency f_i.
2	m	I	1	Input	Number of classes m
3	xmax	{D / R}	1	Input	Upper limit of largest class x_{max}
4	xmin	{D / R}	1	Input	Lower limit of smallest class x_{min}
5	np	I	1	Input	Order of moment to be obtained r When np=1, the linear moment about the origin is obtained.
6	xm	{D∗ / R∗}	1	Output	Moment μ'_1 or μ_r $(r = 2, \cdots, m)$
7	wk	{D∗ / R∗}	m	Work	Work area
8	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) xmax > xmin

(b) m ≥ 1

(c) np ≥ 1

(d) f[i − 1] ≥ 0.0 (i = 1, 2, · · · , m)

(e) $\sum_{i=1}^{m}$ f[i − 1] > 0.0

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
1000	xmax < xmin	xmax and xmin are switched, and processing continues.
3000	xmax = xmin is specified.	Processing is aborted.
3010	Restriction (b) was not satisfied.	
3020	Restriction (c) was not satisfied.	
3030	Restriction (d) was not satisfied.	
3040	Restriction (e) was not satisfied.	

(6) **Notes**

(a) According to the definition, the linear moment about the origin is the mean, and the moment of order 2 about the mean is the variance.

(b) The **skewness** is an indicator representing the degree of asymmetry in the distribution of observed values. The value $\alpha_3 = \frac{\mu_3}{\sqrt{{\mu_2}^3}}$, which is defined using the moments of order 2 and 3 about the mean (μ_2, μ_3) or its square is used as the skewness.

(c) The **kurtosis** is an indicator representing the degree of peakedness in the distribution of observed values. The value $\alpha_4 = \frac{\mu_4}{{\mu_2}^2}$, which is defined using the moments of order 2 and 4 about the mean (μ_2, μ_4) or $\alpha_4 - 3$ is used as the kurtosis.

(7) **Example**

(a) Problem

Assume that the interval $[0, 80]$ is divided into 20 equal-width class intervals and that the observation frequencies of each class $F_1, F_2, \cdots, F_{20}$ are given as follows. Obtain the moment of order 4.

Class	F_i	Class	F_i	Class	F_i	Class	F_i
1	527	6	3942	11	1496	16	448
2	501	7	3737	12	1044	17	283
3	1082	8	3012	13	874	18	260
4	2177	9	2489	14	607	19	207
5	2958	10	1801	15	450	20	144

(b) Input data

Observation frequencies $\{F_i\}$, m = 20, xmax = 80.0, xmin = 0.0 and np = 20.

(c) Main program

```
/*      C interface example for ASL_d2bamo */

#include <stdio.h>
#include <stdlib.h>
#include <asl.h>

int main()
{
    double *f;
    int m;
    double xmax,xmin;
    int np;
    double xm;
    double *wk;
    int ierr;
    int i;
    FILE *fp;

    fp = fopen( "d2bamo.dat", "r" );
    if( fp == NULL )
    {
        printf( "file open error\n" );
        return -1;
    }

    printf( "    *** ASL_d2bamo ***\n" );
    printf( "\n     ** Input **\n\n" );

    fscanf( fp, "%d", &m );
    fscanf( fp, "%lf", &xmax);
    fscanf( fp, "%lf", &xmin);
    fscanf( fp, "%d", &np );

    f = ( double * )malloc((size_t)( sizeof(double) * m ));
    if( f == NULL )
    {
        printf( "no enough memory for array f\n" );
        return -1;
    }
```

```
    wk = ( double * )malloc((size_t)( sizeof(double) * m ));
    if( wk == NULL )
    {
        printf( "no enough memory for array wk\n" );
        return -1;
    }

    printf( "\tm    = %6d\n", m );
    printf( "\txmax = %6.3g\n", xmax );
    printf( "\txmin = %6.3g\n", xmin );
    printf( "\tnp   = %6d\n\n", np );

    printf( "\tFrequencies\n", np );
    for( i=0 ; i<m ; i++ )
    {
        if( i%5 == 0 )
        {
            printf( "\n\t" );
        }
        fscanf( fp, "%lf", &f[i] );
        printf( "%9.5g", f[i] );
    }

    fclose( fp );

    ierr = ASL_d2bamo(f, m, xmax, xmin, np, &xm, wk);

    printf( "\n\n     ** Output **\n\n" );
    printf( "\tierr = %9d\n", ierr );
    printf( "\txm   = %9.4g\n", xm );

    free( f );
    free( wk );

    return 0;
}
```

(d) Output results

```
*** ASL_d2bamo ***

 ** Input **

m    =     20
xmax =     80
xmin =      0
np   =      4

Frequencies

      527      501     1082     2177     2958
     3942     3737     3012     2489     1801
     1496     1044      874      607      450
      448      283      260      207      144

 ** Output **

ierr =         0
xm   = 1.736e+05
```

4.2.6 ASL_d2bahm, ASL_r2bahm Harmonic Mean

(1) **Function**

Given a sample consisting of n observed values $\{x_i\}(i = 1, \cdots, n)$, The ASL_d2bahm or ASL_r2bahm obtains the harmonic mean. Also, the ASL_d2bahm or ASL_r2bahm obtains the harmonic mean when n observed values $\{y_i\}(i = 1, \cdots, n)$ are added.

For a sample consisting of n observed values $\{x_i\}(i = 1, \cdots, n)$, the harmonic mean is defined by the following equations.

Harmonic mean :

$$\mathrm{HM} = \frac{1}{\frac{1}{n}\sum_{i=1}^{n}\frac{1}{x_i}}$$

(2) **Usage**

Double precision:

```
ierr = ASL_d2bahm  (a, n, &ns, &hm, isw);
```

Single precision:

```
ierr = ASL_r2bahm  (a, n, &ns, &hm, isw);
```

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	a	{D∗ / R∗}	n	Input	Sequence of observed values $\{x_i\}$ or $\{y_i\}$
2	n	I	1	Input	Number of observed values n
3	ns	I*	1	Input	Number of observed values before adding observed values (for isw=0 , no initial setting is required)
				Output	Number of observed values n
4	hm	{D∗ / R∗}	1	Input	Harmonic mean before adding observed values (See Note (a)) (for isw=0, no initial setting is required)
				Output	Harmonic mean that was obtained (See Note (a))
5	isw	I	1	Input	Processing switch 0: No previously established basic statistics 1: Added observed values
6	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) isw = 0, 1

(b) n $\geq$ 1

(c) ns $\geq$ 1 (When isw=1)

(d) a[i $-$ 1] $>$ 0.0(i = 1, 2, $\cdots$, n)

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
1000	Restriction (a) was not satisfied.	Processing continues with isw=0.
3000	Restriction (b) was not satisfied.	Processing is aborted.
3010	Restriction (c) was not satisfied.	
3020	Restriction (d) was not satisfied.	
4000	Denominator of harmonic mean was zero.	The absolute value maximum that can be represented is set for the standard deviation.

(6) **Notes**

(a) To obtain the statistics when observed values are added, use the contents of hm and ns, which were calculated before adding the observed values, set the added observed values for a and the number of added observed values for n, set isw to 1, and perform the calculation.

(b) When there are an extremely large number of data values that are widely dispersed, better results are obtained by grouping them into data having absolute values of the same relative size and adding them to the samples in increasing order of size.

(c) Statistics obtained when calculations are performed using an unbiased estimate can be applied to a population for which sampling with replacement is performed from an infinite or finite population. Statistics obtained when calculations are performed using a sample variance can be applied to a population for which the population and sample match.

(7) **Example**

(a) Problem

Given the following sequence of observed values, obtain the harmonic mean and its standard deviation.

$$\{x_i\} = \{300, 600, 150, 30, 20, 120, 200, 100, 50, 40, 50\}$$

Also, obtain the harmonic mean and its standard deviation when the following sequence of observed values are added.

$$\{y_i\} = \{120, 1200, 300, 150, 600, -120, 240, 200, -100, -50\}$$

(b) Input data

First processing :

Sequence of observed values $\{x_i\}$, n = 11 and isw = 0.

Second processing :

Sequence of observed values $\{y_i\}$, n = 10 and isw = 1.

(c) Main program

```
/*      C interface example for ASL_d2bahm */

#include <stdio.h>
#include <stdlib.h>
#include <asl.h>

int main()
{
    double *a;
    int n;
    int ns;
    double hm;
    int isw;
    int ierr;
    int i;
    FILE *fp;

    fp = fopen( "d2bahm.dat", "r" );
    if( fp == NULL )
    {
        printf( "file open error\n" );
        return -1;
    }

    printf( "    *** ASL_d2bahm ***\n" );
    printf( "\n    ** First processing **\n" );
    printf( "\n     ** Input **\n\n" );

    isw=0;
    fscanf( fp, "%d", &n );

    a = ( double * )malloc((size_t)( sizeof(double) * n ));
    if( a == NULL )
    {
        printf( "no enough memory for array a\n" );
        return -1;
    }

    printf( "\tn   = %6d\n", n );
    printf( "\tisw = %6d\n", isw );
    printf( "\n\tObservations\n");

    for( i=0 ; i<n ; i++ )
    {
        if( i%5 == 0 )
        {
            printf( "\n\t" );
        }
        fscanf( fp, "%lf", &a[i] );
        printf( "%9.5g", a[i] );
    }

    ierr = ASL_d2bahm(a, n, &ns, &hm, isw);

    printf( "\n     ** Output **\n\n" );
    printf( "\tierr = %6d\n", ierr );
    printf( "\tns   = %9d\n", ns );
    printf( "\thm   = %9.5g\n", hm );

    printf( "\n\n    ** Continuous processing **\n" );
    printf( "\n     ** Input **\n\n" );

    isw=1;
    fscanf( fp, "%d", &n );

    printf( "\tn   = %6d\n\tns  = %6d\n", n, ns );
    printf( "\tisw = %6d\n", isw );
    printf( "\n\tObservations\n");
    for( i=0 ; i<n ; i++ )
    {
        if( i%5 == 0 )
        {
            printf( "\n\t" );
        }
        fscanf( fp, "%lf", &a[i] );
        printf( "%9.5g", a[i] );
    }

    fclose( fp );

    ierr = ASL_d2bahm(a, n, &ns, &hm, isw);

    printf( "\n\n     ** Output **\n\n" );
    printf( "\tierr = %9d\n", ierr );
    printf( "\tns   = %9d\n", ns );
    printf( "\thm   = %9.5g\n", hm );

    free( a );
```

```
    return 0;
}
```

(d) Output results

```
*** ASL_d2bahm ***

** First processing **

 ** Input **

n   =     11
isw =      0

Observations

      300      600      150       30       20
      120      200      100       50       40
       50
 ** Output **

ierr =      0
ns   =        11
hm   =        60


** Continuous processing **

 ** Input **

n   =     10
ns  =     11
isw =      1

Observations

      120     1200      300      150      600
     -120      240      200     -100      -50

 ** Output **

ierr =         0
ns   =        21
hm   =       120
```

4.2.7 ASL_d2basm, ASL_r2basm Root Mean Square

(1) **Function**

Given a sample consisting of n observed values $\{x_i\}(i = 1, \cdots, n)$, The ASL_d2basm or ASL_r2basm obtains the root mean square Also, the ASL_d2bahm or ASL_r2bahm obtains the root mean square when n observed values $\{y_i\}(i = 1, \cdots, n)$ are added.

For a sample consisting of n observed values $\{x_i\}(i = 1, \cdots, n)$, the root mean square is defined by the following equations.

Root mean square :

$$\mathrm{SM} = \sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}$$

(2) **Usage**

Double precision:

```
ierr = ASL_d2basm  (a, n, &ns, &sm, isw);
```

Single precision:

```
ierr = ASL_r2basm  (a, n, &ns, &sm, isw);
```

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	a	{D∗ / R∗}	n	Input	Sequence of observed values $\{x_i\}$ or $\{y_i\}$
2	n	I	1	Input	Number of observed values n
3	ns	I*	1	Input	Number of observed values before adding observed values (for isw=0, no initial setting is required)
				Output	Number of observed values n
4	sm	{D∗ / R∗}	1	Input	Root mean square before adding observed values (See Note (a)) (for isw=0, no initial setting is required)
				Output	Root mean square that was obtained (See Note (a))
5	isw	I	1	Input	Processing switch 0: No previously established basic statistics 1: Added observed values
6	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) isw = 0, 1

(b) n $\geq$ 1

(c) ns $\geq$ 1 (When isw=1)

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
1000	Restriction (a) was not satisfied.	Processing continues with isw=0.
3000	Restriction (b) was not satisfied.	Processing is aborted.
3010	Restriction (c) was not satisfied.	

(6) **Notes**

(a) To obtain the statistics when observed values are added, use the contents of sm and ns, which were calculated before adding the observed values, set the added observed values for a and the number of added observed values for n, set isw to 1, and perform the calculation.

(b) When there are an extremely large number of data values that are widely dispersed, better results are obtained by grouping them into data having absolute values of the same relative size and adding them to the samples in increasing order of size.

(c) Statistics obtained when calculations are performed using an unbiased estimate can be applied to a population for which sampling with replacement is performed from an infinite or finite population. Statistics obtained when calculations are performed using a sample variance can be applied to a population for which the population and sample match.

(7) **Example**

(a) Problem
Given the following sequence of observed values, obtain the root mean square.

$$\{x_i\} = \{90, 60, 30, 95, 40, 80, 25, 50, 70, 50\}$$

Also, obtain the root mean square when the following sequence of observed values are added.

$$\{y_i\} = \{60, 60, 60, 70, 60, 60, 50, 60, 60, 60\}$$

(b) Input data
First processing :

Sequence of observed values $\{x_i\}$, n = 11 and isw = 0.

Second processing :

Sequence of observed values $\{y_i\}$, n = 10 and isw = 1.

(c) Main program

```
/*      C interface example for ASL_d2basm */

#include <stdio.h>
#include <stdlib.h>
#include <asl.h>

int main()
{
```

```
    double *a;
    int n;
    int ns;
    double sm;
    int isw;
    int ierr;
    int i;
    FILE *fp;

    fp = fopen( "d2basm.dat", "r" );
    if( fp == NULL )
    {
        printf( "file open error\n" );
        return -1;
    }

    printf( "    *** ASL_d2basm ***\n" );
    printf( "\n    ** First processing **\n" );
    printf( "\n     ** Input **\n\n" );

    isw=0;
    fscanf( fp, "%d", &n );

    a = ( double * )malloc((size_t)( sizeof(double) * n ));
    if( a == NULL )
    {
        printf( "no enough memory for array a\n" );
        return -1;
    }

    printf( "\tn   = %6d\n", n );
    printf( "\tisw = %6d\n", isw );
    printf( "\n\tObservations\n");

    for( i=0 ; i<n ; i++ )
    {
        if( i%5 == 0 )
        {
            printf( "\n\t" );
        }
        fscanf( fp, "%lf", &a[i] );
        printf( "%9.5g", a[i] );
    }

    ierr = ASL_d2basm(a, n, &ns, &sm, isw);

    printf( "\n     ** Output **\n\n" );
    printf( "\tierr = %6d\n", ierr );
    printf( "\tns   = %9d\n", ns );
    printf( "\tsm   = %9.5g\n", sm );

    printf( "\n\n    ** Continuous processing **\n" );
    printf( "\n     ** Input **\n\n" );

    isw=1;
    fscanf( fp, "%d", &n );

    printf( "\tn   = %6d\n\tns  = %6d\n", n, ns );
    printf( "\tisw = %6d\n", isw );
    printf( "\n\tObservations\n");
    for( i=0 ; i<n ; i++ )
    {
        if( i%5 == 0 )
        {
            printf( "\n\t" );
        }
        fscanf( fp, "%lf", &a[i] );
        printf( "%9.5g", a[i] );
    }

    fclose( fp );

    ierr = ASL_d2basm(a, n, &ns, &sm, isw);

    printf( "\n\n     ** Output **\n\n" );
    printf( "\tierr = %9d\n", ierr );
    printf( "\tns   = %9d\n", ns );
    printf( "\tsm   = %9.5g\n", sm );

    free( a );

    return 0;
}
```

(d) Output results

```
*** ASL_d2basm ***

** First processing **
```

```
 ** Input **

n   =     11
isw =      0

Observations

       90       60       30       95       40
       80       25       50       70       70
       50
 ** Output **

ierr =      0
ns   =       11
sm   =    63.996


** Continuous processing **

 ** Input **

n   =     10
ns  =     11
isw =      1

Observations

       60       60       60       70       60
       60       50       60       60       60

 ** Output **

ierr =         0
ns   =        21
sm   =    62.202
```

4.3 VARIANCE-COVARIANCE

4.3.1 ASL_d2vcmt, ASL_r2vcmt Variance-Covariance Matrices

(1) **Function**

Given m samples consisting of n observed values $\{x_{ki}\}(k = 1, \cdots, n; i = 1, \cdots, m)$, the ASL_d2vcmt or ASL_r2vcmt obtains the mean of each sample and the variance-covariance among the samples. It also obtains the mean and variance-covariance when n observed values $\{y_{ki}\}$ $(k = 1, \cdots, n; i = 1, \cdots, m)$ are added to each of the m samples for which the mean and variance-covariance are known.

The mean and variance-covariance among the samples for m samples consisting of n observed values $\{x_{ki}\}$ $(k = 1, \cdots, n; i = 1, \cdots, m)$ are defined by the following equations.

Mean :

$$\bar{x}_i = \frac{\sum_{k=1}^{n} x_{ki}}{n} \qquad i = 1, \cdots, m$$

Variance-covariance :

$$d_{ij} = \frac{s_{ij}}{\alpha} \qquad i, j = 1, \cdots, m$$

s_{ij} (sum of squares of deviation matrix) is defined as follows.

$$s_{ij} = \sum_{k=1}^{n} (x_{ki} - \bar{x}_i)(x_{kj} - \bar{x}_j) \qquad i, j = 1, \cdots, m$$

Here, α is n when a sample covariance is used or α is $n - 1$ when an unbiased covariance is used. The diagonal elements of the variance-covariance matrix are the variance values.

(2) **Usage**

Double precision:

```
ierr = ASL_d2vcmt  (a, na, n, m, &ns, x1, d, nd, isw, wk);
```

Single precision:

```
ierr = ASL_r2vcmt  (a, na, n, m, &ns, x1, d, nd, isw, wk);
```

(3) **Arguments and Return Value**

D:Double precision real　Z:Double precision complex　I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real　C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	a	{ D∗ / R∗ }	na×m	Input	Matrix in which observed values are stored (x_{ki}) or (y_{ki}) (See Note (a))
2	na	I	1	Input	Adjustable dimension of array a
3	n	I	1	Input	Number of observed values per sample stored in array a n

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
4	m	I	1	Input	Number of samples m
5	ns	I*	1	Input	Number of observed values per sample before adding observed values (for isw=0 or 2, no initial setting is required)
				Output	Number of observed values per sample n
6	x1	$\left\{\begin{matrix}\text{D}* \\ \text{R}*\end{matrix}\right\}$	m	Input	Mean of each sample before adding observed values (for isw=0 or 2, no initial setting is required)
				Output	Obtained mean of each sample
7	d	$\left\{\begin{matrix}\text{D}* \\ \text{R}*\end{matrix}\right\}$	nd×m	Input	Variance-covariance matrix before adding observed values (See Note (b)) (for isw=0 or 2, no initial setting is required)
				Output	Obtained variance-covariance matrix (See Note (b))
8	nd	I	1	Input	Adjustable dimension of array d
9	isw	I	1	Input	Processing switch 0: Perform calculations using an unbiased covariance (when the result of the previous calculation is not used) 1: Perform calculations using an unbiased covariance (when the result of the previous calculation is used) 2: Perform calculations using a sample covariance (when the result of the previous calculation is not used) 3: Perform calculations using a sample covariance (when the result of the previous calculation is used)
10	wk	$\left\{\begin{matrix}\text{D}* \\ \text{R}*\end{matrix}\right\}$	m	Work	Work area
11	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) isw = 0, 1, 2, 3

(b) na $\geq$ n $\geq$ 1

(c) nd $\geq$ m $\geq$ 1

(d) ns $\geq$ 1 (When isw=1 or 3)

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
1000	Restriction (a) was not satisfied.	Processing continues with isw=0.
1010	An attempt was made to obtain an unbiased covariance when n = 1.	The absolute value maximum that can be represented is set for the covariance.
3000	Restriction (b) or (c) was not satisfied.	Processing is aborted.
3010	Restriction (d) was not satisfied.	

(6) **Notes**

(a) The observed values x_{ki} or y_{ki} $(k = 1, 2, \cdots, n; i = 1, 2, \cdots, m)$ are stored in array a as real matrix (two-dimensional array type) data (See Appendix A).

(b) To obtain the mean and covariance when the same numbers of observed values are added for each sample, use the contents of d and ns, which were calculated before adding the observed values, set the added observed values for a and the number of added observed values for n, set isw to 1 or 3, and perform the calculation. However, when obtaining the covariance, you must set the isw value so that the calculation is performed using a sample covariance following a calculation that used a sample covariance the previous time or so that the calculation is performed using an unbiased covariance following a calculation that used an unbiased covariance the previous time.

(c) When there are an extremely large number of data values that are widely dispersed, better results are obtained by grouping them into data having absolute values of the same relative size and adding them to the samples in increasing order of size.

(d) Statistics obtained when calculations are performed using an unbiased covariance can be applied to a population for which sampling with replacement is performed from an infinite or finite population. Statistics obtained when calculations are performed using a sample covariance can be applied to a population for which the population and sample match.

(7) **Example**

(a) Problem
Obtain the mean of each sample and the covariance among the samples when the observed values are given by matrix X shown below.

$$X = \begin{bmatrix} 7 & 15 & 36 & 61 & 24 \\ 18 & 36 & 43 & 63 & 31 \\ 8 & 11 & 46 & 27 & 15 \\ 6 & 16 & 35 & 64 & 25 \\ 22 & 30 & 40 & 66 & 32 \\ 10 & 11 & 40 & 30 & 18 \\ 17 & 27 & 45 & 55 & 30 \end{bmatrix}$$

Also, obtain the mean of each sample and the covariance among the samples when the observed values given by matrix Y shown below are added.

$$Y = \begin{bmatrix} 15 & 19 & 29 & 57 & 26 \\ 9 & 14 & 31 & 67 & 7 \\ 18 & 18 & 37 & 61 & 20 \end{bmatrix}$$

(b) Input data

First processing :

Observed value matrix X, na $= 100$, n $= 7$, m $= 5$ and isw $= 0$.

Second processing :

Observed value matrix Y, na $= 100$, n $= 3$, m $= 5$ and isw $= 1$.

(c) Main program

```
/*      C interface example for ASL_d2vcmt */

#include <stdio.h>
#include <stdlib.h>
#include <asl.h>

int main()
{
    double *a;
    int na;
    int n;
    int m;
    int ns;
    double *x1;
    double *d;
    int nd;
    int isw;
    double *wk;
    int ierr;
    int i,j;
    FILE *fp;

    fp = fopen( "d2vcmt.dat", "r" );
    if( fp == NULL )
    {
        printf( "file open error\n" );
        return -1;
    }

    printf( "    *** ASL_d2vcmt ***\n" );
    printf( "\n    ** First processing **\n" );
    printf( "\n     ** Input **\n\n" );

    fscanf( fp, "%d", &na );
    fscanf( fp, "%d", &n );
    fscanf( fp, "%d", &m );
    fscanf( fp, "%d", &nd );
    fscanf( fp, "%d", &isw );

    a = ( double * )malloc((size_t)( sizeof(double) * (na*m) ));
    if( a == NULL )
    {
        printf( "no enough memory for array a\n" );
        return -1;
    }

    x1 = ( double * )malloc((size_t)( sizeof(double) * m ));
    if( x1 == NULL )
    {
        printf( "no enough memory for array x1\n" );
        return -1;
    }

    d = ( double * )malloc((size_t)( sizeof(double) * (nd*m) ));
    if( d == NULL )
    {
        printf( "no enough memory for array d\n" );
        return -1;
    }

    wk = ( double * )malloc((size_t)( sizeof(double) * m ));
    if( wk == NULL )
    {
        printf( "no enough memory for array wk\n" );
        return -1;
    }

    printf( "\tna  = %6d\n", na  );
    printf( "\tn   = %6d\n", n   );
    printf( "\tm   = %6d\n", m   );
    printf( "\tnd  = %6d\n", nd  );
    printf( "\tisw = %6d\n", isw );

    printf("\n\tObservations\n\n");
    for( i=0 ; i<n ; i++ )
    {
```

```
    printf("\t");
    for( j=0 ; j<m ; j++ )
    {
        fscanf( fp, "%lf", &a[i+na*j] );
        printf( "%8.3g ", a[i+na*j] );
    }
    printf( "\n" );
}

ierr = ASL_d2vcmt(a, na, n, m, &ns, x1, d, nd, isw, wk);

printf( "\n     ** Output **\n\n" );
printf( "\tierr = %6d\n", ierr );
printf( "\tns   = %6d\n", ns );

printf( "\n\tMean\n\n" );
for( i=0 ; i<m ; i++ )
{
    printf( "\t%8.3g\n", x1[i] );
}

printf( "\n\tCovariance Matrix\n\n" );
for( i=0 ; i<m ; i++ )
{
    printf( "\t" );
    for( j=0 ; j<m ; j++ )
    {
        printf( "%8.3g ", d[i+nd*j] );
    }
    printf( "\n" );
}

printf( "\n\n    ** Continuous processing **\n" );
printf( "\n     ** Input **\n\n" );

fscanf( fp, "%d", &n );
fscanf( fp, "%d", &isw );

printf( "\tna  = %6d\n", na );
printf( "\tn   = %6d\n", n  );
printf( "\tm   = %6d\n", m  );
printf( "\tnd  = %6d\n", nd );
printf( "\tns  = %6d\n", ns );
printf( "\tisw = %6d\n", isw );

printf("\n\tObservations\n\n");
for( i=0 ; i<n ; i++ )
{
    printf("\t");
    for( j=0 ; j<m ; j++ )
    {
        fscanf( fp, "%lf", &a[i+na*j] );
        printf( "%8.3g ", a[i+na*j] );
    }
    printf( "\n" );
}

fclose( fp );

ierr = ASL_d2vcmt(a, na, n, m, &ns, x1, d, nd, isw, wk);

printf( "\n     ** Output **\n\n" );
printf( "\tierr = %6d\n", ierr );
printf( "\tns   = %6d\n", ns );

printf( "\n\tMean\n\n" );
for( i=0 ; i<m ; i++ )
{
    printf( "\t%8.3g\n", x1[i] );
}

printf( "\n\tCovariance Matrix\n\n" );
for( i=0 ; i<m ; i++ )
{
    printf( "\t" );
    for( j=0 ; j<m ; j++ )
    {
        printf( "%8.3g ", d[i+nd*j] );
    }
    printf( "\n" );
}

free( a );
free( x1 );
free( d );
free( wk );
```

```
    return 0;
}
```

(d) Output results

```
*** ASL_d2vcmt ***

** First processing **

 ** Input **

na  =    100
n   =      7
m   =      5
nd  =     10
isw =      0

Observations

       7       15       36       61       24
      18       36       43       63       31
       8       11       46       27       15
       6       16       35       64       25
      22       30       40       66       32
      10       11       40       30       18
      17       27       45       55       30

 ** Output **

ierr =      0
ns   =      7

Mean

    12.6
    20.9
    40.7
    52.3
      25

Covariance Matrix

      40     55.1       11     41.1     31.7
    55.1      100     10.8      111     59.8
      11     10.8     17.9    -33.2    -2.17
    41.1      111    -33.2      277     95.7
    31.7     59.8    -2.17     95.7     43.3


** Continuous processing **

 ** Input **

na  =    100
n   =      3
m   =      5
nd  =     10
ns  =      7
isw =      1

Observations

      15       19       29       57       26
       9       14       31       67        7
      18       18       37       61       20

 ** Output **

ierr =      0
ns   =     10

Mean

      13
    19.7
    38.2
    55.1
    22.8

Covariance Matrix

    31.8     37.8        7     26.8     26.6
    37.8       72       15     62.6     52.2
       7       15     32.2    -39.9     12.6
    26.8     62.6    -39.9      211     36.9
    26.6     52.2     12.6     36.9     62.4
```

4.3.2 ASL_d2vcgr, ASL_r2vcgr Variance-Covariance Matrices (Grouped Data)

(1) **Function**

Given g groups and m samples consisting of n_r observed values for each group $\{x_{ki}^{(r)}\}(k = 1, \cdots, n_r; i = 1, \cdots, m; r = 1, \cdots, g)$, the ASL_d2vcgr or ASL_r2vcgr obtains the mean of each sample in each group, the mean of each sample over all groups, and the covariance among the samples over all groups. It also obtains the mean in each group and the mean and covariance over all groups when n_r observed values $\{y_{ki}^{(r)}\}$ $(k = 1, \cdots, n_r; i = 1, \cdots, m; r = 1, \cdots, g)$ are added in each group to each of the m samples for which the mean of each group and the mean and covariance over all groups are known.

The mean of each group and the mean and covariance over all groups for $\{x_{ki}^{(r)}\}$ $(k = 1, \cdots, n_r; i = 1, \cdots, m; r = 1, \cdots, g)$ are defined by the following equations.

Mean of each group :

$$\bar{x}_i^{(r)} = \frac{\sum_{k=1}^{n_r} x_{ki}^{(r)}}{n_r} \qquad i = 1, \cdots, m; r = 1, \cdots, g$$

Mean over all groups :

$$\bar{x}_i = \frac{\sum_{r=1}^{g} n_r \bar{x}^{(r)}}{\sum_{r=1}^{g} n_r} \qquad i = 1, \cdots, m$$

Covariance over all groups :

$$d_{ij} = \frac{\sum_{r=1}^{g} s_{ij}^{(r)}}{\sum_{r=1}^{g} \alpha_r} \qquad i, j = 1, \cdots, m$$

$s_{ij}^{(r)}$ (deviation sum of product matrix) of each group is defined as follows.

$$s_{ij}^{(r)} = \sum_{k=1}^{n_r} (x_{ki}^{(r)} - \bar{x}_i^{(r)})(x_{kj}^{(r)} - \bar{x}_j^{(r)}) \qquad i, j = 1, \cdots, m$$

Here, α_r is n_r when a sample covariance is used or α_r is $n_r - 1$ when an unbiased covariance is used. The diagonal elements of the covariance matrix are the variance values.

(2) **Usage**

Double precision:

```
ierr = ASL_d2vcgr  (a, na, m, n, k, ns, x1, y, ny, d, nd, isw, wk);
```

Single precision:

```
ierr = ASL_r2vcgr  (a, na, m, n, k, ns, x1, y, ny, d, nd, isw, wk);
```

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	a	{D∗ / R∗}	na×m	Input	Matrix in which observed values are stored ($x_{ki}^{(r)}$) or ($y_{ki}^{(r)}$) (See Note (a))
2	na	I	1	Input	Adjustable dimension of array a
3	m	I	1	Input	Number of samples m
4	n	I*	k	Input	Number of observed values per sample in each group stored in array a n_r
5	k	I	1	Input	Number of groups g
6	ns	I*	k	Input	Number of observed values per sample in each group before adding observed values (for isw=0 or 2, no initial setting is required)
				Output	Number of observed values per sample in each group n_r
7	x1	{D∗ / R∗}	m	Input	Mean of each sample over all groups before adding observed values (for isw=0 or 2, no initial setting is required)
				Output	Obtained mean of each sample over all groups
8	y	{D∗ / R∗}	ny×k	Input	Mean of each sample in each group before adding observed values (for isw=0 or 2, no initial setting is required)
				Output	Obtained mean of each sample in each group
9	ny	I	1	Input	Adjustable dimension of array y
10	d	{D∗ / R∗}	nd×m	Input	Covariance matrix over all groups before adding observed values (See Note (a)) (for isw=0 or 2, no initial setting is required)
				Output	Obtained covariance matrix over all groups (See Note (a))
11	nd	I	1	Input	Adjustable dimension of array d

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
12	isw	I	1	Input	Processing switch 0: Perform calculations using an unbiased covariance (when the result of the previous calculation is not used) 1: Perform calculations using an unbiased covariance (when the result of the previous calculation is used) 2: Perform calculations using a sample covariance (when the result of the previous calculation is not used) 3: Perform calculations using a sample covariance (when the result of the previous calculation is used)
13	wk	$\left\{\begin{matrix}\text{D}* \\ \text{R}*\end{matrix}\right\}$	See Contents	Work	Work area **Size**: $\text{m} \times (\text{m} \times \text{k} + 1)$
14	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) isw = 0, 1, 2, 3

(b) $\text{m} \geq 1$

(c) $\text{k} \geq 1$

(d) $\text{nd} \geq \text{m}$

(e) $\text{ny} \geq \text{m}$

(f) $\text{n}[\text{i}-1] \geq \left\{\begin{matrix} 1 & \text{(When isw} = 0 \text{ or } 2\text{)} \\ 0 & \text{(When isw} = 1 \text{ or } 3\text{)}\end{matrix}\right. \quad (\text{i} = 1, \cdots, \text{k})$

(g) $\text{ns}[\text{i}-1] \geq 1 \quad (\text{i} = 1, \cdots, \text{k})$ (When isw = 1 or 3)

(h) $\text{na} \geq \sum_{\text{i}=1}^{\text{k}} \text{n}[\text{i}-1]$

(i) $\sum_{\text{i}=1}^{\text{k}} \text{n}[\text{i}-1] \geq 1$ (When isw = 1 or 3)

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
1000	Restriction (a) was not satisfied.	Processing continues with isw=0.
1010	An attempt was made to obtain an unbiased covariance when n[0] = ··· = n[k − 1] = 1.	The absolute value maximum that can be represented is set for the covariance.
3000	Any of restriction (b) to (f) were not satisfied.	Processing is aborted.
3010	Restriction (g) was not satisfied.	
3020	Restriction (h) was not satisfied.	
3030	Restriction (i) was not satisfied.	
4000	An error occurred in a lower level function.	

(6) **Notes**

(a) The observed values are stored as follows in array a as real matrix (two-dimensional array type) data (See Appendix A).

$$\begin{bmatrix}
x_{11}^{(1)} & x_{12}^{(1)} & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & x_{1m}^{(1)} \\
x_{21}^{(1)} & x_{22}^{(1)} & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & x_{2m}^{(1)} \\
\vdots & \vdots & & & & & & & & & \vdots \\
x_{n_1 1}^{(1)} & x_{n_1 2}^{(1)} & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & x_{n_1 m}^{(1)} \\
\\
x_{11}^{(2)} & x_{12}^{(2)} & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & x_{1m}^{(2)} \\
x_{21}^{(2)} & x_{22}^{(2)} & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & x_{2m}^{(2)} \\
\vdots & \vdots & & & & & & & & & \vdots \\
\vdots & \vdots & & & & & & & & & \vdots \\
\\
x_{n_2 1}^{(2)} & x_{n_2 2}^{(2)} & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & x_{n_2 m}^{(2)} \\
\\
\vdots & \vdots & & & & & & & & & \vdots \\
x_{11}^{(g)} & x_{12}^{(g)} & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & x_{1m}^{(g)} \\
x_{21}^{(g)} & x_{22}^{(g)} & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & x_{2m}^{(1)} \\
\vdots & \vdots & & & & & & & & & \vdots \\
x_{n_g 1}^{(g)} & x_{n_g 2}^{(g)} & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & x_{n_g m}^{(g)}
\end{bmatrix}$$

(b) To obtain the mean and covariance when the same numbers of observed values are added for each sample in each group, use the contents of ns, y and wk, which were calculated before adding the observed values, set the added observed values for a and the number of added observed values for n, set isw to 1 or 3, and perform the calculation. However, when obtaining the covariance, you must set the isw value so that the calculation is performed using a sample covariance following a calculation that used a sample covariance the previous time or so that the calculation is performed using an unbiased covariance following a calculation that used an unbiased covariance the previous time.

(c) When there are an extremely large number of data values that are widely dispersed, better results are obtained by grouping them into data having absolute values of the same relative size and adding them to the samples in increasing order of size.

(d) Statistics obtained when calculations are performed using an unbiased covariance can be applied to a population for which sampling with replacement is performed from an infinite or finite population. Statistics obtained when calculations are performed using a sample covariance can be applied to a population for which the population and sample match.

(7) **Example**

(a) Problem

When three groups of observed values are given, and the observed values in each group are given by the matrices X_1, X_2 and X_3 shown below, obtain the mean of each sample in each group, the mean of each sample over all groups, and the covariance among the samples over all groups.

$$X_1 = \begin{bmatrix} 10 & 3 & 7 \\ 11 & 5 & 8 \\ 12 & 7 & 6 \end{bmatrix}$$

$$X_2 = \begin{bmatrix} 17 & 12 & 8 \\ 18 & 11 & 6 \\ 18 & 13 & 7 \\ 17 & 11 & 6 \end{bmatrix}$$

$$X_3 = \begin{bmatrix} 11 & 4 & 11 \\ 12 & 6 & 12 \\ 13 & 8 & 10 \\ 15 & 5 & 6 \end{bmatrix}$$

Also, obtain the mean of each sample in each group, the mean of each sample over all groups, and the covariance among the samples over all groups when the observed values given by matrices Y_1 and Y_3 shown below are added to groups 1 and 3.

$$Y_1 = \begin{bmatrix} 14 & 4 & 9 \\ 15 & 6 & 8 \end{bmatrix}$$

$$Y_3 = \begin{bmatrix} 18 & 10 & 13 \\ 14 & 5 & 7 \end{bmatrix}$$

(b) Input data

First processing :

Observed value matrices of each group X_1, X_2 and X_3,

na=100, m=3, k=3,

n[0]=3, n[1]=4, n[2]=4,

ny=10, nd=10 and isw=0.

Second processing :

Observed value matrices of each group Y_1 and Y_3,

na=100, m=3, k=3,

n[0]=2, n[1]=0, n[2]=2,

ny=10, nd=10 and isw=1.

(c) Main program

```
/*      C interface example for ASL_d2vcgr */

#include <stdio.h>
#include <stdlib.h>
#include <asl.h>

int main()
{
    double *a;
    int na;
    int m;
    int *n;
    int k;
    int *ns;
    double *x1;
    double *y;
    int ny;
    double *d;
    int nd;
    int isw;
    double *wk;
    int ierr;
    int i,j,is,ig;
    FILE *fp;

    fp = fopen( "d2vcgr.dat", "r" );
    if( fp == NULL )
    {
        printf( "file open error\n" );
        return -1;
    }

    printf( "    *** ASL_d2vcgr ***\n" );
    printf( "\n    ** First processing **\n" );
    printf( "\n     ** Input **\n\n" );

    fscanf( fp, "%d", &na );
    fscanf( fp, "%d", &m );
    fscanf( fp, "%d", &k );
    fscanf( fp, "%d", &nd );
    fscanf( fp, "%d", &ny );
    fscanf( fp, "%d", &isw );

    a = ( double * )malloc((size_t)( sizeof(double) * (na*m) ));
    if( a == NULL )
    {
        printf( "no enough memory for array a\n" );
        return -1;
    }

    n = ( int * )malloc((size_t)( sizeof(int) * k ));
    if( n == NULL )
    {
        printf( "no enough memory for array n\n" );
        return -1;
    }

    ns = ( int * )malloc((size_t)( sizeof(int) * k ));
    if( ns == NULL )
    {
        printf( "no enough memory for array ns\n" );
        return -1;
    }

    x1 = ( double * )malloc((size_t)( sizeof(double) * m ));
    if( x1 == NULL )
    {
        printf( "no enough memory for array x1\n" );
        return -1;
    }

    y = ( double * )malloc((size_t)( sizeof(double) * (ny*k) ));
    if( y == NULL )
    {
        printf( "no enough memory for array y\n" );
        return -1;
    }

    d = ( double * )malloc((size_t)( sizeof(double) * (nd*m) ));
    if( d == NULL )
    {
        printf( "no enough memory for array d\n" );
        return -1;
    }

    wk = ( double * )malloc((size_t)( sizeof(double) * (m*m*k+m) ));
    if( wk == NULL )
    {
```

```
    printf( "no enough memory for array wk\n" );
    return -1;
}

printf( "\tna  = %6d\n", na  );
printf( "\tm   = %6d\n", m   );
printf( "\tk   = %6d\n", k   );
printf( "\tny  = %6d\n", ny  );
printf( "\tnd  = %6d\n", nd  );
printf( "\tisw = %6d\n", isw );

printf( "\n\tNumber of observations in each group\n\n" );
for( i=0 ; i<k ; i++ )
{
    fscanf( fp, "%d", &n[i] );
    printf("\tn[%3d] = %6d\n",i,n[i]);
}
printf( "\n\tObservations in each group\n\n" );
is = 0;
for( ig=0 ; ig<k ; ig++ )
{
    printf("\tGroup %6d\n",ig);
    for( i=is ; i<is+n[ig] ; i++ )
    {
        printf("\t");
        for( j=0 ; j<m ; j++ )
        {
            fscanf( fp, "%lf", &a[i+na*j] );
            printf( "%8.3g ", a[i+na*j] );
        }
        printf( "\n" );
    }
    is += n[ig];
}

ierr = ASL_d2vcgr(a, na, m, n, k, ns, x1, y, ny, d, nd, isw, wk);

printf( "\n     ** Output **\n\n" );
printf( "\tierr = %6d\n\n", ierr );
for( i=0 ; i<k ; i++ )
{
    printf("\tns[%3d] = %6d\n",i,ns[i]);
}

printf( "\n\tMean over all groups\n\n" );
printf("\t");
for( i=0 ; i<m ; i++ )
{
    printf( "%8.3g ", x1[i] );
}
printf("\n");

printf( "\n\tMean in each group\n\n" );
for( ig=0 ; ig<k ; ig++ )
{
    printf("\tGroup %6d\n",ig);
    printf("\t");
    for( j=0 ; j<m ; j++ )
    {
        printf( "%8.3g ", y[j+ny*ig] );
    }
    printf( "\n" );
}

printf( "\n\tCovariance matrix\n\n" );
for( i=0 ; i<m ; i++ )
{
    printf("\t");
    for( j=0 ; j<m ; j++ )
    {
        printf( "%8.3g ",d[i+nd*j]);
    }
    printf("\n");
}

printf( "\n\n    ** Continuous processing **\n" );
printf( "\n     ** Input **\n\n" );

fscanf( fp, "%d", &isw );

printf( "\tna  = %6d\n", na  );
printf( "\tm   = %6d\n", m   );
printf( "\tk   = %6d\n", k   );
printf( "\tny  = %6d\n", ny  );
printf( "\tnd  = %6d\n", nd  );
printf( "\tisw = %6d\n", isw );
```

```
    printf( "\n\tPre-increased number of observations in each group\n\n" );
    for( i=0 ; i<k ; i++ )
    {
        printf("\tns[%3d] = %6d\n",i,ns[i]);
    }
    printf( "\n\tNumber of observations in each group\n\n" );
    for( i=0 ; i<k ; i++ )
    {
        fscanf( fp, "%d", &n[i] );
        printf("\tn[%3d] = %6d\n",i,n[i]);
    }
    printf( "\n\tObservations in each group\n\n" );
    is = 0;
    for( ig=0 ; ig<k ; ig++ )
    {
        if(n[ig]!=0)
        {
            printf("\tGroup %6d\n",ig);
            for( i=is ; i<is+n[ig] ; i++ )
            {
                printf("\t");
                for( j=0 ; j<m ; j++ )
                {
                    fscanf( fp, "%lf", &a[i+na*j] );
                    printf( "%8.3g ", a[i+na*j] );
                }
                printf( "\n" );
            }
        }
        is += n[ig];
    }

    fclose( fp );

    ierr = ASL_d2vcgr(a, na, m, n, k, ns, x1, y, ny, d, nd, isw, wk);

    printf( "\n     ** Output **\n\n" );
    printf( "\tierr = %6d\n", ierr );
    for( i=0 ; i<k ; i++ )
    {
        printf("\tns[%3d] = %6d\n",i,ns[i]);
    }

    printf( "\n\tMean over all groups\n\n" );
    printf("\t");
    for( i=0 ; i<m ; i++ )
    {
        printf( "%8.3g ", x1[i] );
    }
    printf("\n");

    printf( "\n\tMean in each group\n\n" );
    for( ig=0 ; ig<k ; ig++ )
    {
        printf("\tGroup %6d\n",ig);
        printf("\t");
        for( j=0 ; j<m ; j++ )
        {
            printf( "%8.3g ", y[j+ny*ig] );
        }
        printf( "\n" );
    }

    printf( "\n\tCovariance matrix\n\n" );
    for( i=0 ; i<m ; i++ )
    {
        printf("\t");
        for( j=0 ; j<m ; j++ )
        {
            printf( "%8.3g ",d[i+nd*j]);
        }
        printf("\n");
    }

    free( a );
    free( n );
    free( ns );
    free( x1 );
    free( y );
    free( d );
    free( wk );

    return 0;
}
```

(d) Output results

```
*** ASL_d2vcgr ***

** First processing **

 ** Input **

na  =    100
m   =      3
k   =      3
ny  =     10
nd  =     10
isw =      0

Number of observations in each group

n[  0] =      3
n[  1] =      4
n[  2] =      4

Observations in each group

Group      0
      10         3         7
      11         5         8
      12         7         6
Group      1
      17        12         8
      18        11         6
      18        13         7
      17        11         6
Group      2
      11         4        11
      12         6        12
      13         8        10
      15         5         6

 ** Output **

ierr =      0

ns[  0] =      3
ns[  1] =      4
ns[  2] =      4

Mean over all groups

      14     7.73     7.91

Mean in each group

Group      0
      11         5         7
Group      1
    17.5      11.8     6.75
Group      2
    12.8      5.75     9.75

Covariance matrix

    1.47    0.781    -1.72
   0.781     2.44    0.187
   -1.72    0.187     3.19


** Continuous processing **

 ** Input **

na  =    100
m   =      3
k   =      3
ny  =     10
nd  =     10
isw =      1

Pre-increased number of observations in each group

ns[  0] =      3
ns[  1] =      4
ns[  2] =      4

Number of observations in each group

n[  0] =      2
n[  1] =      0
n[  2] =      2

Observations in each group

Group      0
      14         4         9
      15         6         8
```

```
Group       2
      18        10        13
      14         5         7

 ** Output **

ierr =       0
ns[  0] =       5
ns[  1] =       4
ns[  2] =       6

Mean over all groups

    14.3     7.33     8.27

Mean in each group

Group       0
    12.4         5      7.6
Group       1
    17.5     11.8     6.75
Group       2
    13.8     6.33     9.83

Covariance matrix

    4.09     2.07    0.428
    2.07     3.17     1.34
   0.428     1.34      3.9
```

4.4 CORRELATION COEFFICIENTS

4.4.1 ASL_d2ccmt, ASL_r2ccmt Correlation Matrices

(1) **Function**

Given m samples consisting of n observed values $\boldsymbol{x}_i = \{x_{ki}\}(k = 1, \cdots, n; i = 1, \cdots, m)$, the ASL_d2ccmt or ASL_r2ccmt obtains the mean of each sample and the correlation coefficients between the samples. It also obtains the mean of each sample and the correlation coefficients between the samples when n observed values $\boldsymbol{y}_i = \{y_{ki}\}(k = 1, \cdots, n; i = 1, \cdots, m)$ are added to each of the m samples.
The mean $\bar{x}_i$ of sample $\boldsymbol{x}_i$ $(i = 1, \cdots, m)$ is defined as follows.

$$\bar{x}_i = \frac{t_i}{n}, \qquad i = 1, \cdots, m$$

Here, t_i is the sum, which is defined as follows.

$$t_i = \sum_{k=1}^{n} x_{ki}, \qquad i = 1, \cdots, m$$

The correlation coefficient r_{ij} between samples $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ is defined as follows.

$$r_{ij} = \frac{s_{ij}}{\sqrt{s_{ii} \cdot s_{jj}}}, \qquad i = 1, \cdots, m; j = 1, \cdots, m$$

s_{ij} is defined as follows.

$$s_{ij} = \sum_{k=1}^{n} (x_{ki} - \bar{x}_i)(x_{kj} - \bar{x}_j), \qquad i = 1, \cdots, m; j = 1, \cdots, m$$

The matrix $R = (r_{ij})$ is called the correlation coefficient matrix.

(2) **Usage**

Double precision:

```
ierr = ASL_d2ccmt  (a, na, n, m, &ns, x1, r, nr, isw, wk);
```

Single precision:

```
ierr = ASL_r2ccmt  (a, na, n, m, &ns, x1, r, nr, isw, wk);
```

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	a	{D∗ / R∗}	na×m	Input	Matrix in which observed values are stored (x_{ki}) or (y_{ki}) $(k = 1, \cdots, n;\ i = 1, \cdots, m)$
2	na	I	1	Input	Adjustable dimension of array a
3	n	I	1	Input	Number of observed values per sample stored in array a n
4	m	I	1	Input	Number of samples m
5	ns	I*	1	Input	Number of observed values per sample before adding observed values (for isw=0, no initial setting is required)
				Output	Number of observed values per sample n
6	x1	{D∗ / R∗}	m	Input	Mean of each sample before adding observed values (for isw=0, no initial setting is required)
				Output	Obtained mean of each sample
7	r	{D∗ / R∗}	nr×m	Input	Correlation coefficients between samples before adding observed values (for isw=0, no initial setting is required)
				Output	Obtained correlation coefficients between samples
8	nr	I	1	Input	Adjustable dimension of array r
9	isw	I	1	Input	Processing switch 0: First calculation (when the result of the previous calculation is not used) 1: Add observed values (when the result of the previous calculation is used)
10	wk	{D∗ / R∗}	m	Work	Work area
11	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) isw = 0, 1

(b) na ≥ n ≥ 1

(c) nr ≥ m ≥ 1

(d) ns ≥ 1 (When isw=1)

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
1000	Restriction (a) was not satisfied.	Processing continues with isw=0.
1010	An attempt was made to obtain correlation coefficients when isw=0 and n=1.	The absolute value maximum that can be represented is set for the correlation coefficients.
1020	The variance was zero because all values of a certain sample were equal.	
3000	Restriction (b), (c) was not satisfied.	Processing is aborted.
3010	Restriction (d) was not satisfied.	

(6) **Notes**

(a) To obtain the statistics when the same numbers of observed values are added for each sample, use the contents of x1, r, ns and wk, which were calculated before adding the observed values, set the added observed values for a and the number of added observed values for n, set isw to 1, and perform the calculation.

(b) If a job that had been executed with isw=0 terminated with ierr=1020, the result will not be guaranteed if a job is executed with isw=1 following it.

(c) When there are an extremely large number of data values that are widely dispersed, better results are obtained by grouping them into data having absolute values of the same relative size and adding them to the samples in increasing order of size.

(7) **Example**

(a) Problem

Obtain the mean of each sample and the correlation coefficients between the samples when the observed values are given by matrix X shown below.

$$X = \begin{bmatrix} 7 & 9 & 15 & 60 & 24 \\ 13 & 25 & 13 & 61 & 30 \\ 9 & 24 & 12 & 62 & 31 \\ 7 & 25 & 11 & 63 & 32 \\ 6 & 20 & 15 & 18 & 15 \\ 10 & 30 & 10 & 27 & 17 \\ 7 & 11 & 15 & 60 & 25 \end{bmatrix}$$

Also, obtain the mean of each sample and the correlation coefficients between the samples when the observed values given by matrix Y shown below are added.

$$Y = \begin{bmatrix} 16 & 25 & 13 & 64 & 30 \\ 9 & 26 & 13 & 66 & 32 \\ 8 & 26 & 13 & 66 & 34 \end{bmatrix}$$

(b) Input data

First processing :

Matrix X in which observed values are stored, $\mathrm{na} = 100$, $\mathrm{n} = 7$, $\mathrm{m} = 5$ and $\mathrm{isw} = 0$.

Second processing :

Matrix Y in which observed values are stored, na $= 100$, n $= 3$, m $= 5$ and isw $= 1$.

(c) Main program

```
/*      C interface example for ASL_d2ccmt */

#include <stdio.h>
#include <stdlib.h>
#include <asl.h>

int main()
{
    double *a;
    int na;
    int n;
    int m;
    int ns;
    double *x1;
    double *r;
    int nr;
    int isw;
    double *wk;
    int ierr;
    int i,j;
    FILE *fp;

    fp = fopen( "d2ccmt.dat", "r" );
    if( fp == NULL )
    {
        printf( "file open error\n" );
        return -1;
    }

    printf( "    *** ASL_d2ccmt ***\n" );
    printf( "\n    ** First processing **\n" );
    printf( "\n     ** Input **\n\n" );

    fscanf( fp, "%d", &na );
    fscanf( fp, "%d", &n );
    fscanf( fp, "%d", &m );
    fscanf( fp, "%d", &nr );
    fscanf( fp, "%d", &isw );

    a = ( double * )malloc((size_t)( sizeof(double) * (na*m) ));
    if( a == NULL )
    {
        printf( "no enough memory for array a\n" );
        return -1;
    }

    x1 = ( double * )malloc((size_t)( sizeof(double) * m ));
    if( x1 == NULL )
    {
        printf( "no enough memory for array x1\n" );
        return -1;
    }

    r = ( double * )malloc((size_t)( sizeof(double) * (nr*m) ));
    if( r == NULL )
    {
        printf( "no enough memory for array r\n" );
        return -1;
    }

    wk = ( double * )malloc((size_t)( sizeof(double) * m ));
    if( wk == NULL )
    {
        printf( "no enough memory for array wk\n" );
        return -1;
    }

    printf( "\tna  = %6d\n", na  );
    printf( "\tn   = %6d\n", n   );
    printf( "\tm   = %6d\n", m   );
    printf( "\tnr  = %6d\n", nr  );
    printf( "\tisw = %6d\n", isw );
    printf( "\n\tObservations\n\n");
    for( i=0 ; i<n ; i++ )
    {
        printf("\t");
        for( j=0 ; j<m ; j++ )
        {
            fscanf( fp, "%lf", &a[i+na*j] );
            printf( "%8.3g ", a[i+na*j] );
        }
        printf( "\n" );
    }
```

```
    ierr = ASL_d2ccmt(a, na, n, m, &ns, x1, r, nr, isw, wk);

    printf( "\n     ** Output **\n\n" );
    printf( "\tierr = %6d\n", ierr );
    printf( "\tns   = %6d\n", ns );

    printf( "\n\tCorrelation Matrix\n\n" );
    for( i=0 ; i<m ; i++ )
    {
        printf("\t");
        for( j=0 ; j<m ; j++ )
        {
            printf( "%8.3g ", r[i+nr*j] );
        }
        printf( "\n" );
    }

    printf( "\n\tMean\n\n" );
    for( i=0 ; i<m ; i++ )
    {
        printf( "\t%8.3g\n", x1[i] );
    }
    printf( "\n\n    ** Continuous processing **\n" );
    printf( "\n     ** Input **\n\n" );

    fscanf( fp, "%d", &n );
    fscanf( fp, "%d", &isw );

    printf( "\tna  = %6d\n", na );
    printf( "\tn   = %6d\n", n  );
    printf( "\tm   = %6d\n", m  );
    printf( "\tnr  = %6d\n", nr );
    printf( "\tns  = %6d\n", ns );
    printf( "\tisw = %6d\n", isw );
    printf( "\n\tObservations\n\n");
    for( i=0 ; i<n ; i++ )
    {
        printf("\t");
        for( j=0 ; j<m ; j++ )
        {
            fscanf( fp, "%lf", &a[i+na*j] );
            printf( "%8.3g ", a[i+na*j] );
        }
        printf( "\n" );
    }

    fclose( fp );

    ierr = ASL_d2ccmt(a, na, n, m, &ns, x1, r, nr, isw, wk);

    printf( "\n     ** Output **\n\n" );
    printf( "\tierr = %6d\n", ierr );
    printf( "\tns   = %6d\n", ns );

    printf( "\n\tCorrelation Matrix\n\n" );
    for( i=0 ; i<m ; i++ )
    {
        printf("\t");
        for( j=0 ; j<m ; j++ )
        {
            printf( "%8.3g ", r[i+nr*j] );
        }
        printf( "\n" );
    }

    printf( "\n\tMean\n" );
    for( i=0 ; i<m ; i++ )
    {
        printf( "\t%8.3g\n", x1[i] );
    }

    free( a );
    free( x1 );
    free( r );
    free( wk );

    return 0;
}
```

(d) Output results

```
*** ASL_d2ccmt ***

** First processing **

 ** Input **
```

```
na  =     20
n   =      7
m   =      5
nr  =      5
isw =      0

Observations

       7        9       15       60       24
      13       25       13       61       30
       9       24       12       62       31
       7       25       11       63       32
       6       20       15       18       15
      10       30       10       27       17
       7       11       15       60       25

 ** Output **

ierr =      0
ns   =      7

Correlation Matrix

       1    0.545   -0.427    0.185    0.297
   0.545        1   -0.861   -0.294   0.0491
  -0.427   -0.861        1   0.0294   -0.213
   0.185   -0.294   0.0294        1    0.919
   0.297   0.0491   -0.213    0.919        1

Mean

    8.43
    20.6
      13
    50.1
    24.9


** Continuous processing **

 ** Input **

na  =     20
n   =      3
m   =      5
nr  =      5
ns  =      7
isw =      1

Observations

      16       25       13       64       30
       9       26       13       66       32
       8       26       13       66       34

 ** Output **

ierr =      0
ns   =     10

Correlation Matrix

       1    0.442   -0.272    0.255    0.281
   0.442        1   -0.803   -0.093    0.232
  -0.272   -0.803        1   0.0265   -0.179
   0.255   -0.093   0.0265        1    0.924
   0.281    0.232   -0.179    0.924        1

Mean
     9.2
    22.1
      13
    54.7
      27
```

4.4.2 ASL_d2ccma, ASL_r2ccma Multiple Correlation Coefficients

(1) **Function**

Given m samples consisting of n observed values $\{x_{ki}\}(k = 1, \cdots, n; i = 1, \cdots, m)$, the ASL_d2ccma or ASL_r2ccma obtains the basic statistics (sum, mean, sum of squares of deviation, variance, and standard deviation) and multiple correlation coefficients of each sample. It also obtains the basic statistics and multiple correlation coefficients when n observed values $\{y_{ki}\}$ $(k = 1, \cdots, n; i = 1, \cdots, m)$ are added to each of the m samples for which the basic statistics are known.

The basic statistics and multiple correlation coefficients for m samples consisting of n observed values $\{x_{ki}\}$ $(k = 1, \cdots, n; i = 1, \cdots, m)$ are defined by the following equations.

Sum :

$$t_i = \sum_{k=1}^{n} x_{ki}, \quad i = 1, \cdots, m$$

Mean :

$$\bar{x}_i = \frac{t_i}{n}, \quad i = 1, \cdots, m$$

Sum of squares of deviation :

$$s_i = \sum_{k=1}^{n} (x_{ki} - \bar{x}_i)^2, \quad i = 1, \cdots, m$$

Variance :

$$v_i = \frac{s_i}{\alpha}, \quad i = 1, \cdots, m$$

Standard deviation :

$$d_i = \sqrt{v_i}, \quad i = 1, \cdots, m$$

Here, α is $n - 1$ when an unbiased estimate is used or α is n when a sample variance is used.

Multiple correlation coefficients :

$$r_{i \cdot 1, \cdots, i-1, i+1, \cdots, m} = \sqrt{1 - \frac{\Delta}{\Delta_{ii}}}, \quad i = 1, \cdots, m$$

Here, Δ and Δ_{ij} are the determinant and cofactor matrix of the matrix having the correlation coefficients r_{ij} $(i, j = 1, \cdots, m)$ as elements.

(2) **Usage**

Double precision:

```
ierr = ASL_d2ccma  (a, na, n, m, &ns, stat, r, isw, wk);
```

Single precision:

```
ierr = ASL_r2ccma  (a, na, n, m, &ns, stat, r, isw, wk);
```

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	a	{D∗ / R∗}	na×m	Input	Matrix in which observed values are stored (x_{ki}) or (y_{ki}) (See Note (a))
2	na	I	1	Input	Adjustable dimension of array a
3	n	I	1	Input	Number of observed values per sample stored in array a n
4	m	I	1	Input	Number of samples m
5	ns	I*	1	Input	Number of observed values per sample before adding observed values (for isw=0 or 2, no initial setting is required)
				Output	Number of observed values per sample n
6	stat	{D∗ / R∗}	m×5	Input	Basic statistics before adding observed values (See Note (b)) (for isw=0 or 2, no initial setting is required)
				Output	Obtained basic statistics (See Note (b))
7	r	{D∗ / R∗}	m	Output	Multiple correlation coefficients of each sample
8	isw	I	1	Input	Processing switch 0: Perform calculations using an unbiased estimate (no previously established basic statistics) 1: Perform calculations using an unbiased estimate (added observed values) 2: Perform calculations using a sample variance (no previously established basic statistics) 3: Perform calculations using a sample variance (added observed values)
9	wk	{D∗ / R∗}	See Contents	Work	Work area **Size:**2×m×m+3×m
10	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) isw = 0, 1, 2, 3

(b) na ≥ n ≥ 1

(c) m ≥ 1

(d) ns ≥ 1 (When isw=1 or 3)

(e) n ≥ m (When isw=0 or 2)

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
1000	Restriction (a) was not satisfied.	Processing continues with isw=0.
1010	n = 1 was specified.	When isw = 0, the absolute value maximum that can be represented is set for the variance, standard deviation, and multiple correlation coefficients. When isw = 2, the absolute value maximum that can be represented is set for the multiple correlation coefficients.
1020	The variance was zero because all values of a certain sample were equal.	The absolute value maximum that can be represented is set for the multiple correlation coefficients.
3000	Restriction (b), (c) was not satisfied.	Processing is aborted.
3010	Restriction (d) was not satisfied.	
3020	Restriction (e) was not satisfied.	
4000	The inverse matrix of the simple correlation matrix was not obtained.	

(6) **Notes**

(a) The observed values x_{ki} or y_{ki} $(k = 1, 2, \cdots, n; i = 1, 2, \cdots, m)$ are stored in array a as real matrix (two-dimensional array type) data (See Appendix A).

(b) The basic statistics are stored as follows in the array stat.

stat$[i-1]$	:	Sum t_i
stat$[(i-1)+m]$	:	Mean $\bar{x}_i$
stat$[(i-1)+m\times 2]$	:	Sum of squares of deviation s_i , i = $1, \cdots,$ m
stat$[(i-1)+m\times 3]$	:	Variance v_i
stat$[(i-1)+m\times 4]$	:	Standard deviation d_i

(c) To obtain the basic statistics and multiple correlation coefficients when the same numbers of observed values are added for each sample, use the contents of stat, ns and wk, which were calculated before adding the observed values, set the added observed values for a and the number of added observed values for n, set isw to 1 or 3, and perform the calculation. However, when obtaining the variance or standard deviation, you must set the isw value so that the calculation is performed using a sample variance following a calculation that used a sample variance the previous time or so that the calculation is performed using an unbiased estimate following a calculation that used an unbiased estimate the previous time.

(d) If a job that had been executed with isw=0 or isw=2 terminated with ierr=1020, the result will not be guaranteed if a job is executed with isw=1 or isw=3 following it.

(e) When there are an extremely large number of data values that are widely dispersed, better results are obtained by grouping them into data having absolute values of the same relative size and adding them to the samples in increasing order of size.

(f) Statistics obtained when calculations are performed using an unbiased estimate can be applied to a population for which sampling with replacement is performed from an infinite or finite population. Statistics obtained when calculations are performed using a sample variance can be applied to a population for which the population and sample match.

(7) **Example**

(a) Problem

Obtain the basic statistics (sum, mean, sum of squares of deviation, variance, and standard deviation) and multiple correlation coefficients for each sample when the observed values are given by matrix X shown below.

$$X = \begin{bmatrix} 23.9 & 64.6 & 2.41 \\ 21.4 & 65.2 & 2.14 \\ 23.6 & 57.7 & 2.61 \\ 23.2 & 61.0 & 2.24 \\ 25.0 & 86.5 & 2.78 \\ 25.7 & 88.8 & 2.95 \\ 23.2 & 91.0 & 2.91 \end{bmatrix}$$

Also, obtain the basic statistics (sum, mean, sum of squares of deviation, variance, and standard deviation) and multiple correlation coefficients for each sample when the observed values given by matrix Y shown below are added.

$$Y = \begin{bmatrix} 24.3 & 84.6 & 3.12 \\ 25.3 & 89.2 & 3.01 \\ 26.5 & 90.8 & 2.73 \end{bmatrix}$$

(b) Input data

First processing :

Matrix X in which observed values are stored, $\mathrm{na} = 100$, $\mathrm{n} = 7$, $\mathrm{m} = 3$ and $\mathrm{isw} = 0$.

Second processing :

Matrix Y in which observed values are stored, $\mathrm{na} = 100$, $\mathrm{n} = 3$, $\mathrm{m} = 3$ and $\mathrm{isw} = 1$.

(c) Main program

```
/*      C interface example for ASL_d2ccma */

#include <stdio.h>
#include <stdlib.h>
#include <asl.h>

int main()
{
    double *a;
    int na;
    int n;
    int m;
    int ns;
    double *stat;
    double *r;
    int isw;
    double *wk;
    int ierr;
    int i,j;
    FILE *fp;

    fp = fopen( "d2ccma.dat", "r" );
    if( fp == NULL )
    {
        printf( "file open error\n" );
        return -1;
    }
```

```
printf( "    *** ASL_d2ccma ***\n" );
printf( "\n    ** First processing **\n" );
printf( "\n     ** Input **\n\n" );

fscanf( fp, "%d", &na );
fscanf( fp, "%d", &n );
fscanf( fp, "%d", &m );
fscanf( fp, "%d", &isw );

a = ( double * )malloc((size_t)( sizeof(double) * (na*m) ));
if( a == NULL )
{
    printf( "no enough memory for array a\n" );
    return -1;
}

stat = ( double * )malloc((size_t)( sizeof(double) * (m*5) ));
if( stat == NULL )
{
    printf( "no enough memory for array stat\n" );
    return -1;
}

r = ( double * )malloc((size_t)( sizeof(double) * m ));
if( r == NULL )
{
    printf( "no enough memory for array r\n" );
    return -1;
}

wk = ( double * )malloc((size_t)( sizeof(double) * (2*m*m+3*m) ));
if( wk == NULL )
{
    printf( "no enough memory for array wk\n" );
    return -1;
}

printf( "\tna  = %6d\n", na );
printf( "\tn   = %6d\n", n  );
printf( "\tm   = %6d\n", m  );
printf( "\tisw = %6d\n", isw );

printf("\n\tObservations\n\n");
for( i=0 ; i<n ; i++ )
{
    printf("\t");
    for( j=0 ; j<m ; j++ )
    {
        fscanf( fp, "%lf", &a[i+na*j] );
        printf( "%8.3g ", a[i+na*j] );
    }
    printf( "\n" );
}

ierr = ASL_d2ccma(a, na, n, m, &ns, stat, r, isw, wk);

printf( "\n     ** Output **\n\n" );
printf( "\tierr = %6d\n", ierr );
printf( "\tns   = %6d\n\n", ns );

printf( "\t    Sum       Mean      Sum of    Variance    Standard \n" );
printf( "\t                        Squares               deviation\n" );
printf( "\t-------------------------------------------------------\n" );
for( i=0 ; i<m ; i++ )
{
    printf("\t");
    for( j=0 ; j<5 ; j++ )
    {
        printf( "%8.3g   ", stat[i+m*j] );
    }
    printf( "\n" );
}

printf( "\n\tMultiple Correlation Coefficient\n\n" );
for( i=0 ; i<m ; i++ )
{
    printf( "\t%8.3g\n", r[i] );
}

printf( "\n    ** Continuous processing **\n" );
printf( "\n     ** Input **\n\n" );

fscanf( fp, "%d", &n );
fscanf( fp, "%d", &isw );

printf( "\tna  = %6d\n", na  );
printf( "\tn   = %6d\n", n   );
printf( "\tm   = %6d\n", m   );
```

```
    printf( "\tns  = %6d\n", ns  );
    printf( "\tisw = %6d\n", isw );

    printf("\n\tObservations\n\n");
    for( i=0 ; i<n ; i++ )
    {
        printf("\t");
        for( j=0 ; j<m ; j++ )
        {
            fscanf( fp, "%lf", &a[i+na*j] );
            printf( "%8.3g ", a[i+na*j] );
        }
        printf( "\n" );
    }

    fclose( fp );

    ierr = ASL_d2ccma(a, na, n, m, &ns, stat, r, isw, wk);

    printf( "\n     ** Output **\n\n" );
    printf( "\tierr = %6d\n", ierr );
    printf( "\tns   = %6d\n\n", ns );

    printf( "\t    Sum       Mean      Sum of    Variance    Standard \n" );
    printf( "\t                        Squares               diviation\n" );
    printf( "\t-------------------------------------------------------\n" );
    for( i=0 ; i<m ; i++ )
    {
        printf("\t");
        for( j=0 ; j<5 ; j++ )
        {
            printf( "%8.3g   ", stat[i+m*j] );
        }
        printf( "\n" );
    }

    printf( "\n\tMultiple Correlation Coefficient\n\n" );
    for( i=0 ; i<m ; i++ )
    {
        printf( "\t%8.3g\n", r[i] );
    }

    free( a );
    free( stat );
    free( r );
    free( wk );

    return 0;
}
```

(d) Output results

```
*** ASL_d2ccma ***

** First processing **

 ** Input **

na  =    100
n   =      7
m   =      3
isw =      0

Observations

    23.9     64.6     2.41
    21.4     65.2     2.14
    23.6     57.7     2.61
    23.2       61     2.24
      25     86.5     2.78
    25.7     88.8     2.95
    23.2       91     2.91

 ** Output **

ierr =      0
ns   =      7

    Sum       Mean      Sum of    Variance    Standard
                        Squares               deviation
-----------------------------------------------------
     166       23.7       11.5       1.92       1.39
     515       73.5   1.26e+03        211       14.5
      18       2.58      0.625      0.104      0.323

Multiple Correlation Coefficient

   0.749
   0.825
   0.893
```

```
** Continuous processing **

 ** Input **

na  =    100
n   =      3
m   =      3
ns  =      7
isw =      1

Observations

    24.3     84.6     3.12
    25.3     89.2     3.01
    26.5     90.8     2.73

 ** Output **

ierr =      0
ns   =     10

    Sum        Mean       Sum of     Variance     Standard
                          Squares                 diviation
-------------------------------------------------------------
     242        24.2         19.7         2.19        1.48
     779        77.9    1.74e+03          193        13.9
    26.9        2.69            1        0.111       0.334

Multiple Correlation Coefficient

   0.685
   0.823
   0.816
```

4.4.3 ASL_d2ccpr, ASL_r2ccpr Partial Correlation Coefficients

(1) **Function**

Given m samples consisting of n observed values $\{x_{ki}\}(k = 1, \cdots, n; i = 1, \cdots, m)$, the ASL_d2ccpr or ASL_r2ccpr obtains the basic statistics (sum, mean, sum of squares of deviation, variance, and standard deviation) and partial correlation coefficients of each sample. It also obtains the basic statistics and partial correlation coefficients when n observed values $\{y_{ki}\}$ $(k = 1, \cdots, n; i = 1, \cdots, m)$ are added to each of the m samples for which the basic statistics are known.

The basic statistics and partial correlation coefficients for m samples consisting of n observed values $\{x_{ki}\}$ $(k = 1, \cdots, n; i = 1, \cdots, m)$ are defined by the following equations.

Sum :

$$t_i = \sum_{k=1}^{n} x_{ki}, \quad i = 1, \cdots, m$$

Mean :

$$\bar{x}_i = \frac{t_i}{n}, \quad i = 1, \cdots, m$$

Sum of squares of deviation :

$$s_i = \sum_{k=1}^{n} (x_{ki} - \bar{x}_i)^2, \quad i = 1, \cdots, m$$

Variance :

$$v_i = \frac{s_i}{\alpha}, \quad i = 1, \cdots, m$$

Standard deviation :

$$d_i = \sqrt{v_i}, \quad i = 1, \cdots, m$$

Here, α is $n - 1$ when an unbiased estimate is used or α is n when a sample variance is used.

Partial correlation coefficients :

$$r_{i,j \cdot 1, \cdots, i-1, i+1, \cdots j-1, j+1, \cdots m} = -\frac{\Delta_{ij}}{\sqrt{\Delta_{ii}\Delta_{jj}}} \quad i, j = 1, \cdots, m$$

Here, Δ_{ij} is the cofactor matrix of the matrix having the correlation coefficients r_{ij} $(i, j = 1, \cdots, m)$ as elements.

(2) **Usage**

Double precision:

```
ierr = ASL_d2ccpr  (a, na, n, m, &ns, stat, r, nr, isw, wk);
```

Single precision:

```
ierr = ASL_r2ccpr  (a, na, n, m, &ns, stat, r, nr, isw, wk);
```

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	a	{D∗ / R∗}	na×m	Input	Matrix in which observed values are stored (x_{ki}) or (y_{ki})(See Note (a))
2	na	I	1	Input	Adjustable dimension of array a
3	n	I	1	Input	Number of observed values per sample stored in array a n
4	m	I	1	Input	Number of samples m
5	ns	I*	1	Input	Number of observed values per sample before adding observed values (for isw=0 or 2, no initial setting is required)
				Output	Number of observed values per sample n
6	stat	{D∗ / R∗}	m×5	Input	Basic statistics before adding observed values (See Note (b)) (for isw=0 or 2, no initial setting is required)
				Output	Obtained basic statistics (See Note (b))
7	r	{D∗ / R∗}	nr×m	Output	Partial correlation coefficients between samples $r_{i,j\cdot 1,\cdots,i-1,i+1,\cdots j-1,j+1,\cdots m}$ (See Note (a))
8	nr	I	1	Input	Adjustable dimension of array r
9	isw	I	1	Input	Processing switch 0: Perform calculations using an unbiased estimate (no previously established basic statistics) 1: Perform calculations using an unbiased estimate (added observed values) 2: Perform calculations using a sample variance (no previously established basic statistics) 3: Perform calculations using a sample variance (added observed values)
10	wk	{D∗ / R∗}	See Contents	Work	Work area 2×m×m+3×m
11	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) isw = 0, 1, 2, 3

(b) na $\geq$ n $\geq$ 1

(c) nr $\geq$ m $\geq$ 1

(d) ns $\geq$ 1 (When isw=1 or 3)

(e) n $\geq$ m (When isw=0 or 2)

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
1000	Restriction (a) was not satisfied.	Processing continues with isw=0.
1010	n = 1 was specified.	When isw = 0, the absolute value maximum that can be represented is set for the variance, standard deviation, and partial correlation coefficients. When isw = 2, the absolute value maximum that can be represented is set for the partial correlation coefficients.
1020	The variance was zero because all values of a certain sample were equal.	The absolute value maximum that can be represented is set for the partial correlation coefficients.
3000	Restriction (b), (c) was not satisfied.	Processing is aborted.
3010	Restriction (d) was not satisfied.	
3020	Restriction (e) was not satisfied.	
4000	The inverse matrix of the simple correlation matrix was not obtained.	

(6) **Notes**

(a) The observed values x_{ki} or y_{ki} $(k = 1, 2, \cdots, n; i = 1, 2, \cdots, m)$ are stored in array a as real matrix (two-dimensional array type) data.
Also, the partial correlation coefficients $\hat{r}_{ij} = r_{i,j\cdot 1,\cdots,i-1,i+1,\cdots j-1,j+1,\cdots m}$ $(i, j = 1, 2, \cdots, m)$ are stored in array r as real matrix (two-dimensional array type) data.
For the method of array data storage, see Appendix A.

(b) The basic statistics are stored as follows in the array stat.

stat$[i-1]$	:	Sum t_i
stat$[(i-1)+m]$	:	Mean $\bar{x}_i$
stat$[(i-1)+m\times 2]$	:	Sum of squares of deviation s_i , i = $1, \cdots,$ m
stat$[(i-1)+m\times 3]$	:	Variance v_i
stat$[(i-1)+m\times 4]$	:	Standard deviation d_i

(c) To obtain the basic statistics and partial correlation coefficients when the same numbers of observed values are added for each sample, use the contents of stat, ns and wk, which were calculated before adding the observed values, set the added observed values for a and the number of added observed values for n, set isw to 1 or 3, and perform the calculation. However, when obtaining the variance or

standard deviation, you must set the isw value so that the calculation is performed using a sample variance following a calculation that used a sample variance the previous time or so that the calculation is performed using an unbiased estimate following a calculation that used an unbiased estimate the previous time.

(d) If a job that had been executed with isw=0 or isw=2 terminated with ierr=1020, the result will not be guaranteed if a job is executed with isw=1 or isw=3 following it.

(e) When there are an extremely large number of data values that are widely dispersed, better results are obtained by grouping them into data having absolute values of the same relative size and adding them to the samples in increasing order of size.

(f) Statistics obtained when calculations are performed using an unbiased estimate can be applied to a population for which sampling with replacement is performed from an infinite or finite population. Statistics obtained when calculations are performed using a sample variance can be applied to a population for which the population and sample match.

(7) **Example**

(a) Problem

Obtain the basic statistics (sum, mean, sum of squares of deviation, variance, and standard deviation) and partial correlation coefficients for each sample when the observed values are given by matrix X shown below.

$$X = \begin{bmatrix} 84 & 58 & 42 \\ 92 & 88 & 86 \\ 88 & 80 & 98 \\ 66 & 72 & 64 \\ 64 & 50 & 40 \\ 80 & 94 & 74 \\ 90 & 92 & 94 \end{bmatrix}$$

Also, obtain the basic statistics (sum, mean, sum of squares of deviation, variance, and standard deviation) and partial correlation coefficients for each sample when the observed values given by matrix Y shown below are added.

$$Y = \begin{bmatrix} 40 & 36 & 8 \\ 78 & 50 & 86 \\ 82 & 62 & 66 \end{bmatrix}$$

(b) Input data

First processing :

Matrix X in which observed values are stored, $\mathrm{na} = 100$, $\mathrm{n} = 7$, $\mathrm{m} = 3$ and $\mathrm{isw} = 0$.

Second processing :

Matrix Y in which observed values are stored, $\mathrm{na} = 100$, $\mathrm{n} = 3$, $\mathrm{m} = 3$ and $\mathrm{isw} = 1$.

(c) Main program

```
/*      C interface example for ASL_d2ccpr */

#include <stdio.h>
#include <stdlib.h>
#include <asl.h>

int main()
{
```

```
double *a;
int na;
int n;
int m;
int ns;
double *stat;
double *r;
int nr;
int isw;
double *wk;
int ierr;
int i,j;
FILE *fp;

fp = fopen( "d2ccpr.dat", "r" );
if( fp == NULL )
{
    printf( "file open error\n" );
    return -1;
}

printf( "    *** ASL_d2ccpr ***\n" );
printf( "\n    ** First processing **\n" );
printf( "\n     ** Input **\n\n" );

fscanf( fp, "%d", &na );
fscanf( fp, "%d", &n );
fscanf( fp, "%d", &m );
fscanf( fp, "%d", &nr );
fscanf( fp, "%d", &isw );

a = ( double * )malloc((size_t)( sizeof(double) * (na*m) ));
if( a == NULL )
{
    printf( "no enough memory for array a\n" );
    return -1;
}

stat = ( double * )malloc((size_t)( sizeof(double) * (m*5) ));
if( stat == NULL )
{
    printf( "no enough memory for array stat\n" );
    return -1;
}

r = ( double * )malloc((size_t)( sizeof(double) * (nr*m) ));
if( r == NULL )
{
    printf( "no enough memory for array r\n" );
    return -1;
}

wk = ( double * )malloc((size_t)( sizeof(double) * (2*m*m+3*m) ));
if( wk == NULL )
{
    printf( "no enough memory for array wk\n" );
    return -1;
}

printf( "\tna  = %6d\n", na  );
printf( "\tn   = %6d\n", n   );
printf( "\tm   = %6d\n", m   );
printf( "\tnr  = %6d\n", nr  );
printf( "\tisw = %6d\n", isw );

printf( "\n\tObservations\n\n" );
for( i=0 ; i<n ; i++ )
{
    printf("\t");
    for( j=0 ; j<m ; j++ )
    {
        fscanf( fp, "%lf", &a[i+na*j] );
        printf( "%8.3g ", a[i+na*j] );
    }
    printf( "\n" );
}

ierr = ASL_d2ccpr(a, na, n, m, &ns, stat, r, nr, isw, wk);

printf( "\n     ** Output **\n\n" );
printf( "\tierr = %6d\n", ierr );
printf( "\tns   = %6d\n\n", ns );

printf( "\t    Sum       Mean      Sum of    Variance    Standard \n" );
printf( "\t                        Squares               diviation\n" );
printf( "\t-------------------------------------------------------\n" );
for( i=0 ; i<m ; i++ )
{
    printf("\t");
    for( j=0 ; j<5 ; j++ )
```

```
        {
            printf( "%8.3g   ", stat[i+m*j] );
        }
        printf( "\n" );
    }

    printf( "\n\tPartial Correlation Coefficient Matrix\n\n" );
    for( i=0 ; i<m ; i++ )
    {
        printf( "\t" );
        for( j=0 ; j<m ; j++ )
        {
            printf( "%8.3g ", r[i+nr*j] );
        }
        printf( "\n" );
    }

    printf( "\n\n    ** Continuous processing **\n" );
    printf( "\n     ** Input **\n\n" );

    fscanf( fp, "%d", &n );
    fscanf( fp, "%d", &isw );

    printf( "\tna  = %6d\n", na  );
    printf( "\tn   = %6d\n", n   );
    printf( "\tm   = %6d\n", m   );
    printf( "\tnr  = %6d\n", nr  );
    printf( "\tns  = %6d\n", ns  );
    printf( "\tisw = %6d\n", isw );

    printf( "\n\tObservations\n\n" );
    for( i=0 ; i<n ; i++ )
    {
        printf("\t");
        for( j=0 ; j<m ; j++ )
        {
            fscanf( fp, "%lf", &a[i+na*j] );
            printf( "%8.3g ", a[i+na*j] );
        }
        printf( "\n" );
    }

    fclose( fp );

    ierr = ASL_d2ccpr(a, na, n, m, &ns, stat, r, nr, isw, wk);

    printf( "\n     ** Output **\n\n" );
    printf( "\tierr = %6d\n", ierr );
    printf( "\tns   = %6d\n\n", ns );

    printf( "\t    Sum       Mean      Sum of    Variance    Standard \n" );
    printf( "\t                        Squares               diviation\n" );
    printf( "\t--------------------------------------------------------\n" );
    for( i=0 ; i<m ; i++ )
    {
        printf("\t");
        for( j=0 ; j<5 ; j++ )
        {
            printf( "%8.3g   ", stat[i+m*j] );
        }
        printf( "\n" );
    }

    printf( "\n\tPartial Correlation Coefficient Matrix\n\n" );
    for( i=0 ; i<m ; i++ )
    {
        printf( "\t" );
        for( j=0 ; j<m ; j++ )
        {
            printf( "%8.3g ", r[i+nr*j] );
        }
        printf( "\n" );
    }

    free( a );
    free( stat );
    free( r );
    free( wk );

    return 0;
}
```

(d) Output results

```
*** ASL_d2ccpr ***

** First processing **
```

```
 ** Input **

na  =    100
n   =      7
m   =      3
nr  =      3
isw =      0

Observations

      84       58       42
      92       88       86
      88       80       98
      66       72       64
      64       50       40
      80       94       74
      90       92       94

 ** Output **

ierr =      0
ns   =      7

    Sum       Mean      Sum of    Variance    Standard
                        Squares               diviation
-------------------------------------------------------
     564       80.6         774         129        11.4
     534       76.3    1.76e+03         293        17.1
     498       71.1    3.34e+03         557        23.6

Partial Correlation Coefficient Matrix

      -1     0.12    0.366
    0.12       -1    0.744
   0.366    0.744       -1


** Continuous processing **

 ** Input **

na  =    100
n   =      3
m   =      3
nr  =      3
ns  =      7
isw =      1

Observations

      40       36        8
      78       50       86
      82       62       66

 ** Output **

ierr =      0
ns   =     10

    Sum       Mean      Sum of    Variance    Standard
                        Squares               diviation
-------------------------------------------------------
     764       76.4    2.25e+03         250        15.8
     682       68.2    3.62e+03         402        20.1
     658       65.8    7.29e+03         810        28.5

Partial Correlation Coefficient Matrix

      -1    0.273    0.643
   0.273       -1    0.378
   0.643    0.378       -1
```

Chapter 5

TIME SERIES ANALYSIS

5.1 INTRODUCTION

This library provides the following functions for performing time series analysis.

- Autocovariance and Cross Covariance
- Autocorrelation and Cross Correlation
- Smoothing and Demand Forecasting

5.1.1 Explanation

(1) **Basic Statistics of Time Series Data**

Let time series data be represented by $x_1, x_2, \cdots, x_n$. Let the mean of the first and last $n-l$ data values among this time series data be represented by $\mu^{(l)}$ and $\nu^{(l)}$ as follows, where $(l = 0, 1, \cdots, m-1; m \leq n)$.

$$\mu^{(l)} = \frac{\sum_{i=1}^{n-l} x_i}{(n-l)} \ (l = 0, 1, \cdots, m-1)$$

$$\nu^{(l)} = \frac{\sum_{i=1}^{n-l} x_{i+l}}{(n-l)} \ (l = 0, 1, \cdots, m-1)$$

The **autocovariance** $c^{(l)}$ at this time is defined as follows.

$$c^{(l)} = \frac{\sum_{i=1}^{n-l} (x_i - \mu^{(l)})(x_{i+l} - \nu^{(l)})}{(n-l)} \ (l = 0, 1, \cdots, m-1)$$

$c^{(l)}$ can also be expressed as follows.

$$c^{(l)} = \frac{\sum_{j=l+1}^{n} (x_j - \nu^{(l)})(x_{j-l} - \mu^{(l)})}{(n-l)} \ (l = 0, 1, \cdots, m-1)$$

If the sample variance (not an unbiased estimate) of the first and last $n-l$ data values among this time series data are represented by $u^{(l)}$ and $v^{(l)}$ as follows:

$$u^{(l)} = \frac{\sum_{i=1}^{n-l} (x_i - \mu^{(l)})^2}{(n-l)} \ (l = 0, 1, \cdots, m-1)$$

$$v^{(l)} = \frac{\sum_{i=1}^{n-l} (x_{i+l} - \nu^{(l)})^2}{(n-l)} \ (l = 0, 1, \cdots, m-1)$$

the **autocorrelation coefficient** $r^{(l)}$ is defined as follows.

$$\begin{aligned} r^{(l)} &= \frac{c^{(l)}}{\sqrt{u^{(l)} v^{(l)}}} \\ &= \frac{\sum_{i=1}^{n-l} (x_i - \mu^{(l)})(x_{i+l} - \nu^{(l)})}{\sqrt{\sum_{i=1}^{n-l} (x_i - \mu^{(l)})^2} \sqrt{\sum_{i=1}^{n-l} (x_{i+l} - \nu^{(l)})^2}} \end{aligned}$$

The variable l is called the **lag**.

Now, let two sets of time series data be represented by $x_1, x_2, \cdots, x_n$ and $y_1, y_2, \cdots, y_n$. Let the mean of

the first and last $n-l$ data values among each of these sets of time series data be represented by ${\mu_x}^{(l)}$ and ${\nu_x}^{(l)}$ for x_i and ${\mu_y}^{(l)}$ and ${\nu_y}^{(l)}$ for y_i $(i=1,2,\cdots,n)$ as follows, where $(l=0,1,\cdots,m-1; m \leq n)$.

$$
{\mu_x}^{(l)} = \frac{\sum_{i=1}^{n-l} x_i}{(n-l)} \ (l=0,1,\cdots,m-1)
$$

$$
{\nu_x}^{(l)} = \frac{\sum_{i=1}^{n-l} x_{i+l}}{(n-l)} \ (l=0,1,\cdots,m-1)
$$

$$
{\mu_y}^{(l)} = \frac{\sum_{i=1}^{n-l} y_i}{(n-l)} \ (l=0,1,\cdots,m-1)
$$

$$
{\nu_y}^{(l)} = \frac{\sum_{i=1}^{n-l} y_{i+l}}{(n-l)} \ (l=0,1,\cdots,m-1)
$$

The **cross covariances** ${c_{xy}}^{(l)}$ and ${c_{yx}}^{(l)}$ at this time are defined as follows.

$$
{c_{xy}}^{(l)} = \frac{\sum_{i=1}^{n-l} (x_i - {\mu_x}^{(l)})(y_{i+l} - {\nu_y}^{(l)})}{(n-l)} \ (l=0,1,\cdots,m-1)
$$

$$
{c_{yx}}^{(l)} = \frac{\sum_{i=1}^{n-l} (y_i - {\mu_y}^{(l)})(x_{i+l} - {\nu_x}^{(l)})}{(n-l)} \ (l=0,1,\cdots,m-1)
$$

If the sample variances (not unbiased estimates) of the first and last $n-l$ data values among each of these sets of time series data are represented by ${u_x}^{(l)}$ and ${v_x}^{(l)}$ for x_i and ${u_y}^{(l)}$ and ${v_y}^{(l)}$ for y_i $(i=1,2,\cdots,n)$ as follows:

$$
{u_x}^{(l)} = \frac{\sum_{i=1}^{n-l} (x_i - {\mu_x}^{(l)})^2}{(n-l)} \ (l=0,1,\cdots,m-1)
$$

$$
{v_x}^{(l)} = \frac{\sum_{i=1}^{n-l} (x_{i+l} - {\nu_x}^{(l)})^2}{(n-l)} \ (l=0,1,\cdots,m-1)
$$

$$
{u_y}^{(l)} = \frac{\sum_{i=1}^{n-l} (y_i - {\mu_y}^{(l)})^2}{(n-l)} \ (l=0,1,\cdots,m-1)
$$

$$
{v_y}^{(l)} = \frac{\sum_{i=1}^{n-l} (y_{i+l} - {\nu_y}^{(l)})^2}{(n-l)} \ (l=0,1,\cdots,m-1)
$$

the **cross correlation coefficients** ${r_{xy}}^{(l)}$ and ${r_{yx}}^{(l)}$ are defined as follows.

$$
\begin{aligned}
{r_{xy}}^{(l)} &= \frac{{c_{xy}}^{(l)}}{\sqrt{{u_x}^{(l)}{v_y}^{(l)}}} \\
&= \frac{\sum_{i=1}^{n-l}(x_i - {\mu_x}^{(l)})(y_{i+l} - {\nu_y}^{(l)})}{\sqrt{\sum_{i=1}^{n-l}(x_i - {\mu_x}^{(l)})^2}\sqrt{\sum_{i=1}^{n-l}(y_{i+l} - {\nu_y}^{(l)})^2}}
\end{aligned}
$$

$$
\begin{aligned}
{r_{yx}}^{(l)} &= \frac{{c_{yx}}^{(l)}}{\sqrt{{u_y}^{(l)}{v_x}^{(l)}}} \\
&= \frac{\sum_{i=1}^{n-l}(y_i - {\mu_y}^{(l)})(x_{i+l} - {\nu_x}^{(l)})}{\sqrt{\sum_{i=1}^{n-l}(y_i - {\mu_y}^{(l)})^2}\sqrt{\sum_{i=1}^{n-l}(x_{i+l} - {\nu_x}^{(l)})^2}}
\end{aligned}
$$

(2) **Smoothing and Demand Forecasting**

(a) Moving Averages

Let n given time series data values be represented by:

$$x_1, x_2, \cdots, x_n$$

and m specified weights be represented by:

$$w_1, w_2, \cdots, w_m$$

The weighted moving average M_k^w is defined as follows.

$$M_k^w = \sum_{j=1}^{m} \frac{(x_{k+j-1} \cdot w_j)}{\sum_{j=1}^{m} w_j} \ (k = 1, 2, \cdots, n - m + 1)$$

m can be called the smoothing bandwidth. Usually, the weight coefficients w_j are determined so that the following relationship is satisfied.

$$\sum_{j=1}^{m} w_j = 1$$

(b) Single Exponential Smoothing

The single exponential smoothing equation for a given time series $\cdots, x_{n-1}, x_n$ (x_n is the most recent data) is defined as follows.

$$S_t = \alpha x_t + (1 - \alpha)S_{t-1} \ (t = M, M + 1, \cdots, n; M \to -\infty)$$

Here, S_t is the smoothing value of x_t and α is the smoothing constant. Now, assume the forecasting model has the following kind of structure.

$$x_t = a + \varepsilon_t$$

Here, a is a constant, and ε_t is an error term that independently obeys $N(0, \sigma^2)$. At this time, the mathematical expectation value E_t at time t and the forecast value E_{t+L} at L periods after t are as follows.

$$E_{t+L} = E_t = S_t$$

(c) Double Exponential Smoothing

The double exponential smoothing equation for a given time series $\cdots, x_{n-1}, x_n$ (x_n is the most recent data) is defined as follows.

$$S_t = \alpha x_t + (1-\alpha)S_{t-1}$$

$$D_t = \alpha S_t + (1-\alpha)D_{t-1} \ (t = M, M+1, \cdots, n; M \to -\infty)$$

Here, S_t is the single exponential smoothing value of x_t, D_t is the double exponential smoothing value of x_t, and α is the smoothing constant. Now, assume the forecasting model has the following kind of structure.

$$x_t = a + bt + \varepsilon_t$$

Here, a and b are constants, and ε_t is an error term that independently obeys $N(0, \sigma^2)$. At this time, the mathematical expectation value E_t at time t and the forecast value E_{t+L} at L periods after t are as follows.

$$E_t = a + bt$$

$$S_t = a + bt - \frac{1-\alpha}{\alpha}b = E_t - \frac{1-\alpha}{\alpha} \cdot b$$

$$D_t = a + bt - \frac{2(1-\alpha)}{\alpha}b = E_t - \frac{2(1-\alpha)}{\alpha} \cdot b$$

Therefore, the following relationships hold:

$$E_t = 2S_t - D_t$$

$$b = B_t = \frac{\alpha}{1-\alpha}(S_t - D_t)$$

Also:

$$E_{t+L} = (2S_t - D_t) + B_t L$$

Here, B_t is called the **linear trend estimate**.

(d) Triple Exponential Smoothing

The triple exponential smoothing equation for a given time series $\cdots, x_{n-1}, x_n$ (x_n is the most recent data) is defined as follows.

$$S_t = \alpha x_t + (1-\alpha)S_{t-1}$$

$$D_t = \alpha S_t + (1-\alpha)D_{t-1}$$

$$T_t = \alpha D_t + (1-\alpha)T_{t-1} \ (t = M, M+1, \cdots, n; M \to -\infty)$$

Here, S_t is the single exponential smoothing value of x_t, D_t is the double exponential smoothing value of x_t, T_t is the triple exponential smoothing value of x_t, and α is the smoothing constant. Now, assume the forecasting model has the following kind of structure.

$$x_t = a + bt + \frac{c}{2}t^2 + \varepsilon_t$$

Here, a, b, and c are constants, and ε_t is an error term that independently obeys $N(0, \sigma^2)$. At this time, the mathematical expectation value E_t at time t and the forecast value E_{t+L} at L periods after t are as follows.

$$E_t = a + bt + \frac{c}{2}t^2$$

$$S_t = a + bt + \frac{c}{2}t^2 - (b + ct)\frac{1-\alpha}{\alpha} + \frac{c}{2} \cdot \frac{(1-\alpha)(1+(1-\alpha))}{(\alpha)^2}$$

$$D_t = a + bt + \frac{c}{2}t^2 - (b + ct)\frac{2(1-\alpha)}{\alpha} + \frac{c}{2} \cdot \frac{(1-\alpha)(2+2(1-\alpha))}{(\alpha)^2}$$

$$T_t = a + bt + \frac{c}{2}t^2 - (b + ct)\frac{3(1-\alpha)}{\alpha} + \frac{c}{2} \cdot \frac{(1-\alpha)(3+3(1-\alpha))}{(\alpha)^2}$$

Therefore, the following relationships hold:

$$E_t = 3S_t - 3D_t + T_t$$

$$b = B_t = \frac{\alpha}{2(1-\alpha)^2}\{(6-5\alpha)S_t - 2(5-4\alpha)D_t + (4-3\alpha)T_t\}$$

$$c = C_t = \frac{(\alpha)^2}{(1-\alpha)^2}(S_t - 2D_t + T_t)$$

Also:

$$E_{t+L} = E_t + B_t L + \frac{C_t}{2}L^2$$

Here, B_t is called the **linear trend estimate** and C_t is called the **quadratic trend estimate**.

5.1.2 Reference Bibliography

(1) Brigham, E. Oran, "The Fast Fourier Transform", Prentice-Hall Inc. , (1974).

5.2 AUTOCOVARIANCE AND CROSS COVARIANCE

5.2.1 ASL_dfcvsc, ASL_rfcvsc Autocovariances

(1) **Function**

Let time series data be represented by $x_1, x_2, \cdots, x_n$. The ASL_dfcvsc or ASL_rfcvsc obtains the sum, mean, sum of squares of deviation, and standard deviation (unbiased estimate), which are defined by the following equations, for the first $n-l$ $(l = 1, 2, \cdots, m; m \leq n)$ data values among this time series data.

Sum:

$$s^{(l)} = \sum_{i=1}^{n-l} x_i \ (l = 1, 2, \cdots, m)$$

Mean:

$$\mu^{(l)} = \frac{\sum_{i=1}^{n-l} x_i}{(n-l)} \ (l = 1, 2, \cdots, m)$$

Sum of squares of deviation:

$$v^{(l)} = \sum_{i=1}^{n-l} (x_i - \mu^{(l)})^2 \ (l = 1, 2, \cdots, m)$$

Standard deviation (unbiased estimate):

$$\sigma^{(l)} = \sqrt{\frac{v^{(l)}}{(n-l-1)}}$$

The function also obtains the autocovariance, which is defined by the following equation.

$$c^{(l)} = \frac{\sum_{i=1}^{n-l} (x_i - \mu^{(l)})(x_{i+l} - \nu^{(l)})}{(n-l)} \ (l = 0, 1, \cdots, m-1)$$

Here, $\nu^{(l)}$ is the mean of the last $n-l$ data values, which is defined as follows.

$$\nu^{(l)} = \frac{\sum_{i=1}^{n-l} x_{i+l}}{(n-l)} \ (l = 1, 2, \cdots, m)$$

(2) **Usage**

Double precision:

```
ierr = ASL_dfcvsc  (a, n, m, v, stat);
```

Single precision:

```
ierr = ASL_rfcvsc  (a, n, m, v, stat);
```

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	a	{D∗ / R∗}	n	Input	Time series data x_i
2	n	I	1	Input	Number of time series data n
3	m	I	1	Input	Number of autocovariances to be obtained m
4	v	{D∗ / R∗}	m	Output	Autocovariance $c^{(l)}$
5	stat	{D∗ / R∗}	m×4	Output	Basic statistics (See Note (a))
6	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) n ≥ 2

(b) 0 < m ≤ n

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
3000	Restriction (a) or (b) was not satisfied.	Processing is aborted.

(6) **Notes**

(a) The sum $s^{(l)}$, mean $\mu^{(l)}$, sum of squares of deviation $v^{(l)}$, and standard deviation $\sigma^{(l)}$ $(l = 1, 2, \cdots, m)$ are output as follows in array stat as a real matrix (two-dimensional array type) (See Appendix A).

$$\begin{bmatrix} s^{(1)} & \mu^{(1)} & v^{(1)} & \sigma^{(1)} \\ s^{(2)} & \mu^{(2)} & v^{(2)} & \sigma^{(2)} \\ \vdots & \vdots & \vdots & \vdots \\ s^{(m)} & \mu^{(m)} & v^{(m)} & \sigma^{(m)} \end{bmatrix}$$

(7) **Example**

(a) Problem

Obtain autocovariances when the following time series data a is given.

a[0] = 48.1, a[12] = 26.1
a[1] = 54.7, a[13] = 22.3
a[2] = 58.0, a[14] = 24.1
a[3] = 64.2, a[15] = 33.1
a[4] = 56.1, a[16] = 42.3
a[5] = 44.9, a[17] = 52.5

a[6] = 36.1, a[18] = 36.0
a[7] = 34.2, a[19] = 23.5
a[8] = 50.1, a[20] = 14.9
a[9] = 54.9, a[21] = 19.8
a[10] = 55.1, a[22] = 25.0
a[11] = 48.4, a[23] = 35.4

(b) Input data

Time series data a, n=24 and number of autocovariances to be obtained m=14.

(c) Main program

```
/*      C interface example for ASL_dfcvsc */

#include <stdio.h>
#include <stdlib.h>
#include <asl.h>

int main()
{
    double *a, *v;
    int n, m;
    double *stat;
    int ierr;

    int i;

    FILE *fp;


    n=24;
    m=14;

    printf( "    *** ASL_dfcvsc ***\n" );
    printf( "\n     ** Input **\n\n" );
    printf( "\tn=%3d\n", n );
    printf( "\tm=%3d\n\n", m );

    a = ( double * )malloc((size_t)( sizeof(double) * (n) ));
    if( a == NULL )
    {
        printf( "no enough memory for array a\n" );
        return -1;
    }

    v = ( double * )malloc((size_t)( sizeof(double) * (m) ));
    if( v == NULL )
    {
        printf( "no enough memory for array v\n" );
        return -1;
    }

    stat = ( double * )malloc((size_t)( sizeof(double) * (m*4) ));
    if( stat == NULL )
    {
        printf( "no enough memory for array stat\n" );
        return -1;
    }

    fp = fopen( "dfcvsc.dat", "r" );
    if( fp == NULL )
    {
        printf( "file open error\n" );
        return -1;
    }

    printf( "\tTime series data(Array a)" );
    for( i=0 ; i<n ; i++ )
    {
        fscanf( fp, "%lf", &a[i] );
        if( i%6 == 0 )
        {
            printf( "\n\t" );
        }
        printf( "%8.3g", a[i] );
    }
    printf( "\n" );

    fclose( fp );

    ierr = ASL_dfcvsc(a, n, m, v, stat);

    printf( "\n     ** Output **\n\n" );
    printf( "\tierr = %6d\n\n", ierr );
```

```
    printf( "\tSum" );
    for( i=0 ; i<m ; i++ )
    {
        if( i%6 == 0 )
        {
            printf( "\n\t" );
        }
        printf( "%8.3g", stat[i] );
    }
    printf( "\n\n" );

    printf( "\tMean" );
    for( i=0 ; i<m ; i++ )
    {
        if( i%6 == 0 )
        {
            printf( "\n\t" );
        }
        printf( "%8.3g", stat[i+m] );
    }
    printf( "\n\n" );

    printf( "\tSum of squares of deviations" );
    for( i=0 ; i<m ; i++ )
    {
        if( i%6 == 0 )
        {
            printf( "\n\t" );
        }
        printf( "%8.4g", stat[i+m*2] );
    }
    printf( "\n\n" );

    printf( "\tStandard deviation" );
    for( i=0 ; i<m ; i++ )
    {
        if( i%6 == 0 )
        {
            printf( "\n\t" );
        }
        printf( "%8.3g", stat[i+m*3] );
    }
    printf( "\n\n" );

    printf( "\tAutocovariance" );
    for( i=0 ; i<m ; i++ )
    {
        if( i%6 == 0 )
        {
            printf( "\n\t" );
        }
        printf( "%8.4g", v[i] );
    }
    printf( "\n" );

    free( a );
    free( v );
    free( stat );

    return 0;
}
```

(d) Output results

```
*** ASL_dfcvsc ***

 ** Input **

n= 24
m= 14

Time series data(Array a)
    48.1    54.7      58    64.2    56.1    44.9
    36.1    34.2    50.1    54.9    55.1    48.4
    26.1    22.3    24.1    33.1    42.3    52.5
      36    23.5    14.9    19.8      25    35.4

 ** Output **

ierr =      0

Sum
     960     924     899     880     865     841
     805     753     710     677     653     631
     605     556

Mean
      40    40.2    40.9    41.9    43.2    44.3
    44.7    44.3    44.4    45.2    46.7    48.5
    50.4    50.6
```

```
Sum of squares of deviations
    4687    4665    4424    3958    3193    2783
    2711    2647    2643    2507    2032    1393
   848.1   843.7

Standard deviation
    14.3    14.6    14.5    14.1      13    12.4
    12.6    12.9    13.3    13.4    12.5    10.8
    8.78    9.19

Autocovariance
    4687    3579    1540  -232.7  -621.5   448.1
    1945    2523    1891   480.5  -673.3  -878.1
    -136   624.6
```

5.2.2 ASL_dfcvcs, ASL_rfcvcs Cross Covariances

(1) **Function**

Let two sets of time series data be represented as follows.

$$x_1, x_2, \cdots, x_n$$

$$y_1, y_2, \cdots, y_n$$

The ASL_dfcvcs or ASL_rfcvcs obtains the cross covariances ${c_{xy}}^{(l)}$ and ${c_{yx}}^{(l)}$ $(l = 1, 2, \cdots, m; m \leq n)$, which are defined by the following equations.

$${c_{xy}}^{(l)} = \frac{\sum_{i=1}^{n-l}(x_i - {\mu_x}^{(l)})(y_{i+l} - {\nu_y}^{(l)})}{(n-l)} \ (l = 1, 2, \cdots, m)$$

$${c_{yx}}^{(l)} = \frac{\sum_{i=1}^{n-l}(y_i - {\mu_y}^{(l)})(x_{i+l} - {\nu_x}^{(l)})}{(n-l)} \ (l = 1, 2, \cdots, m)$$

Here, ${\mu_x}^{(l)}$, ${\nu_x}^{(l)}$, ${\mu_y}^{(l)}$ and ${\nu_y}^{(l)}$, which represent the means for x_i and y_i $(i = 1, 2, \cdots, n)$ of the first and last $n - l$ data values, respectively, among these sets of time series data where $(l = 1, 2, \cdots, m; m \leq n)$, are defined as follows.

$${\mu_x}^{(l)} = \frac{\sum_{i=1}^{n-l} x_i}{(n-l)} \ (l = 1, 2, \cdots, m)$$

$${\nu_x}^{(l)} = \frac{\sum_{i=1}^{n-l} x_{i+l}}{(n-l)} \ (l = 1, 2, \cdots, m)$$

$${\mu_y}^{(l)} = \frac{\sum_{i=1}^{n-l} y_i}{(n-l)} \ (l = 1, 2, \cdots, m)$$

$${\nu_y}^{(l)} = \frac{\sum_{i=1}^{n-l} y_{i+l}}{(n-l)} \ (l = 1, 2, \cdots, m)$$

(2) **Usage**

Double precision:

```
ierr = ASL_dfcvcs  (x, n, y, m, vx, vy);
```

Single precision:

```
ierr = ASL_rfcvcs  (x, n, y, m, vx, vy);
```

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	x	{D∗ / R∗}	n	Input	Time series data x_i
2	n	I	1	Input	Number of time series data n
3	y	{D∗ / R∗}	n	Input	Time series data y_i
4	m	I	1	Input	Number of cross covariances to be obtained m
5	vx	{D∗ / R∗}	m	Output	Cross covariance ${c_{xy}}^{(l)}$
6	vy	{D∗ / R∗}	m	Output	Cross covariance ${c_{yx}}^{(l)}$
7	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) n ≥ 2

(b) 0 < m ≤ n

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
3000	Restriction (a) or (b) was not satisfied.	Processing is aborted.

(6) **Notes**

None

(7) **Example**

(a) Problem

Obtain cross covariances when the following two sets of time series data x and y are given.
x[0] = 2.05, y[0] = 22.23
x[1] = 2.10, y[1] = 22.34
x[2] = 3.40, y[2] = 22.30
x[3] = 5.01, y[3] = 21.90
x[4] = 7.35, y[4] = 21.31
x[5] = 9.43, y[5] = 20.41
x[6] = 10.82, y[6] = 19.43
x[7] = 11.09, y[7] = 18.41
x[8] = 10.41, y[8] = 17.72

x[9] = 7.85, y[9] = 17.81
x[10] = 3.97, y[10] = 18.01
x[11] = 2.90, y[11] = 18.72
x[12] = 2.85, y[12] = 19.51
x[13] = 3.50, y[13] = 20.39
x[14] = 4.97, y[14] = 21.54
x[15] = 6.77, y[15] = 22.33
x[16] = 9.61, y[16] = 22.35
x[17] = 11.91, y[17] = 21.69
x[18] = 12.81, y[18] = 19.97
x[19] = 10.92, y[19] = 19.02
x[20] = 9.30, y[20] = 18.73
x[21] = 6.13, y[21] = 18.91
x[22] = 4.54, y[22] = 19.34
x[23] = 4.72, y[23] = 19.94

(b) Input data

Time series data x and y, n=24 and number of cross covariances to be obtained m=14.

(c) Main program

```
/*      C interface example for ASL_dfcvcs */

#include <stdio.h>
#include <stdlib.h>
#include <asl.h>

int main()
{
    double *x, *y;
    int n, m;
    double *vx, *vy;
    int ierr;

    int i;

    FILE *fp;


    n=24;
    m=12;

    printf( "    *** ASL_dfcvcs ***\n" );
    printf( "\n     ** Input **\n\n" );
    printf( "\tn=%3d\n", n );
    printf( "\tm=%3d\n\n", m );

    x = ( double * )malloc((size_t)( sizeof(double) * (n) ));
    if( x == NULL )
    {
        printf( "no enough memory for array x\n" );
        return -1;
    }

    y = ( double * )malloc((size_t)( sizeof(double) * (n) ));
    if( y == NULL )
    {
        printf( "no enough memory for array y\n" );
        return -1;
    }

    vx = ( double * )malloc((size_t)( sizeof(double) * (m) ));
    if( vx == NULL )
    {
        printf( "no enough memory for array vx\n" );
        return -1;
    }

    vy = ( double * )malloc((size_t)( sizeof(double) * (m) ));
    if( vy == NULL )
    {
        printf( "no enough memory for array vy\n" );
        return -1;
    }

    fp = fopen( "dfcvcs.dat", "r" );
```

```
    if( fp == NULL )
    {
        printf( "file open error\n" );
        return -1;
    }

    for( i=0 ; i<n ; i++ )
    {
        fscanf( fp, "%lf", &x[i] );
    }

    for( i=0 ; i<n ; i++ )
    {
        fscanf( fp, "%lf", &y[i] );
    }

    fclose( fp );

    printf( "\t  data x    data y\n" );
    for( i=0 ; i<n ; i++ )
    {
        printf( "\t%8.3g  %8.3g\n", x[i],y[i] );
    }


    ierr = ASL_dfcvcs(x, n, y, m, vx, vy);

    printf( "\n     ** Output **\n\n" );
    printf( "\tierr = %6d\n\n", ierr );

    printf( "\tCross covariance\n" );
    printf( "\t    vx        vy\n" );
    for( i=0 ; i<m ; i++ )
    {
        printf( "\t%8.3g  %8.3g\n", vx[i],vy[i] );
    }
    printf( "\n" );

    free( x );
    free( y );
    free( vx );
    free( vy );

    return 0;
}
```

(d) Output results

```
*** ASL_dfcvcs ***

 ** Input **

n= 24
m= 12

  data x    data y
    2.05      22.2
     2.1      22.3
     3.4      22.3
    5.01      21.9
    7.35      21.3
    9.43      20.4
    10.8      19.4
    11.1      18.4
    10.4      17.7
    7.85      17.8
    3.97        18
     2.9      18.7
    2.85      19.5
     3.5      20.4
    4.97      21.5
    6.77      22.3
    9.61      22.4
    11.9      21.7
    12.8        20
    10.9        19
     9.3      18.7
    6.13      18.9
    4.54      19.3
    4.72      19.9

 ** Output **

ierr =      0

Cross covariance
    vx        vy
   -32.3     -32.3
   -72.6      22.6
   -93.6      66.4
     -90      86.1
     -66      78.1
```

```
-27.9      48.6
 13.2      9.81
 47.8     -26.5
 67.5     -54.7
 67.4     -70.3
   50     -69.5
 21.2       -50
```

5.3 AUTOCORRELATION AND CROSS CORRELATION

5.3.1 ASL_dfcrsc, ASL_rfcrsc
Autocorrelation Coefficients

(1) **Function**

Let time series data be represented by $x_1, x_2, \cdots, x_n$. The ASL_dfcrsc or ASL_rfcrsc obtains the sum, mean, sum of squares of deviation, and standard deviation (unbiased estimate), which are defined by the following equations, for the first $n-l$ $(l = 1, 2, \cdots, m; m \leq n)$ data values among this time series data.

Sum:

$$s^{(l)} = \sum_{i=1}^{n-l} x_i \ (l = 1, 2, \cdots, m)$$

Mean:

$$\mu^{(l)} = \frac{\sum_{i=1}^{n-l} x_i}{(n-l)} \ (l = 1, 2, \cdots, m)$$

Sum of squares of deviation:

$$v^{(l)} = \sum_{i=1}^{n-l} (x_i - \mu^{(l)})^2 \ (l = 1, 2, \cdots, m)$$

Standard deviation (unbiased estimate):

$$\sigma^{(l)} = \sqrt{\frac{v^{(l)}}{(n-l-1)}}$$

The function also obtains the autocorrelation coefficient, which is defined by the following equation.

$$r^{(l)} = \frac{\sum_{i=1}^{n-l} (x_i - \mu^{(l)})(x_{i+l} - \nu^{(l)})}{\sqrt{\sum_{i=1}^{n-l} (x_i - \mu^{(l)})^2} \sqrt{\sum_{i=1}^{n-l} (x_{i+l} - \nu^{(l)})^2}}$$

Here, $\nu^{(l)}$ is the mean of the last $n-l$ data values, which is defined as follows.

$$\nu^{(l)} = \frac{\sum_{i=1}^{n-l} x_{i+l}}{(n-l)} \ (l = 1, 2, \cdots, m)$$

(2) **Usage**

Double precision:

ierr = ASL_dfcrsc (a, n, m, r, stat);

Single precision:

ierr = ASL_rfcrsc (a, n, m, r, stat);

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	a	{D∗ / R∗}	n	Input	Time series data x_i
2	n	I	1	Input	Number of time series data n
3	m	I	1	Input	Number of autocorrelation coefficients to be obtained m
4	r	{D∗ / R∗}	m	Output	Autocorrelation coefficient $r^{(l)}$
5	stat	{D∗ / R∗}	m×4	Output	Basic statistics (See Note (a))
6	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) n ≥ 2

(b) 1 ≤ m ≤n

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
1000	For some l, the standard deviation $\sigma^{(l)}$ or correlation coefficient $r^{(l)}$ was not obtained.	0.0 is set for the portion corresponding to $\sigma^{(l)}$ or $r^{(l)}$.
3000	Restriction (a) was not satisfied.	Processing is aborted.
3100	Restriction (b) was not satisfied.	

(6) **Notes**

(a) The sum $s^{(l)}$, mean $\mu^{(l)}$, sum of squares of deviation $v^{(l)}$, and standard deviation $\sigma^{(l)}$ $(l = 1, 2, \cdots, m)$ are output as follows in array stat as a real matrix (two-dimensional array type) (See Appendix A).

$$\begin{bmatrix} s^{(1)} & \mu^{(1)} & v^{(1)} & \sigma^{(1)} \\ s^{(2)} & \mu^{(2)} & v^{(2)} & \sigma^{(2)} \\ \vdots & \vdots & \vdots & \vdots \\ s^{(m)} & \mu^{(m)} & v^{(m)} & \sigma^{(m)} \end{bmatrix}$$

(b) When ierr=1000, the sum $s^{(l)}$, mean $\mu^{(l)}$, and sum of squares of deviation $v^{(l)}$ are obtained, and 0.0 is set in the positions where the standard deviation $\sigma^{(l)}$ or correlation coefficient $r^{(l)}$ was not obtained $(l = 1, 2, \cdots, m)$.

5.3.2 ASL_dfcrcz, ASL_rfcrcz
Cross Correlation Coefficients (Zero Means)

(1) **Function**

Given the following two sets of time series data for which the means are zero:

$$x_1, x_2, \cdots, x_n$$

$$y_1, y_2, \cdots, y_n$$

the ASL_dfcrcz or ASL_rfcrcz obtains an approximation ${r_{xy}}^{(l)}$ $(l = 1, 2, \cdots, m)$ of the cross correlation coefficient, which is defined by the following equation.

$${r_{xy}}^{(l)} = \frac{n}{n-l} \frac{\sum_{i=1}^{n-l} x_i y_{i+l}}{\sqrt{\sum_{i=1}^{n} (x_i)^2} \sqrt{\sum_{i=1}^{n} (y_i)^2}} \quad (l = 1, 2, \cdots, m; n \gg m)$$

(2) **Usage**

Double precision:

ierr = ASL_dfcrcz (x, n, y, m, crr);

Single precision:

ierr = ASL_rfcrcz (x, n, y, m, crr);

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	x	{D∗ / R∗}	n	Input	Time series data x_i
2	n	I	1	Input	Number observed values n
3	y	{D∗ / R∗}	n	Input	Time series data y_i
4	m	I	1	Input	Number of cross correlation coefficients to be obtained m
5	crr	{D∗ / R∗}	m	Output	Cross correlation coefficient approximation ${r_{xy}}^{(l)}$
6	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) $1 \leq \mathrm{m} \leq \mathrm{n}$

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
1000	The sum of squares of x_i or sum of squares of y_i was smaller than the unit for determining error.	${r_{xy}}^{(l)} = 0.0$ was set for all l.
1100	For some l the cross correlation coefficient ${r_{xy}}^{(l)}$ was larger than 1.	Processing continues.
3000	Restriction (a) was not satisfied.	Processing is aborted.

(6) **Notes**

(a) The function 5.3.3 $\left\{\begin{matrix}\text{ASL\_dfcrcs}\\\text{ASL\_rfcrcs}\end{matrix}\right\}$ must be used for time series data for which the mean is not zero.

(b) When ierr=1100, the correlation coefficients ${r_{xy}}^{(l)}$ are all calculated. However, since the correlation coefficient definition is not satisfied for those having a value greater than or equal to 1, they cannot be used as correlation coefficients.

5.3.3 ASL_dfcrcs, ASL_rfcrcs Cross Correlation Coefficients

(1) **Function**

Let two sets of time series data be represented as follows.

$$x_1, x_2, \cdots, x_n$$

$$y_1, y_2, \cdots, y_n$$

The ASL_dfcrcs or ASL_rfcrcs obtains the cross correlation coefficients ${r_{xy}}^{(l)}$ and ${r_{yx}}^{(l)}$ $(l = 1, 2, \cdots, m; m \leq n)$, which are defined by the following equations.

$${r_{xy}}^{(l)} = \frac{\sum_{i=1}^{n-l}(x_i - {\mu_x}^{(l)})(y_{i+l} - {\nu_y}^{(l)})}{\sqrt{\sum_{i=1}^{n-l}(x_i - {\mu_x}^{(l)})^2}\sqrt{\sum_{i=1}^{n-l}(y_{i+l} - {\nu_y}^{(l)})^2}} \ (l = 1, 2, \cdots, m)$$

$${r_{yx}}^{(l)} = \frac{\sum_{i=1}^{n-l}(y_i - {\mu_y}^{(l)})(x_{i+l} - {\nu_x}^{(l)})}{\sqrt{\sum_{i=1}^{n-l}(y_i - {\mu_y}^{(l)})^2}\sqrt{\sum_{i=1}^{n-l}(x_{i+l} - {\nu_x}^{(l)})^2}} \ (l = 1, 2, \cdots, m)$$

Here, ${\mu_x}^{(l)}$, ${\nu_x}^{(l)}$, ${\mu_y}^{(l)}$, and ${\nu_y}^{(l)}$, which represent the means for x_i and y_i $(i = 1, 2, \cdots, n)$ of the first and last $n - l$ data values, respectively, among these sets of time series data where $(l = 1, 2, \cdots, m; m \leq n)$, are defined as follows.

$${\mu_x}^{(l)} = \frac{\sum_{i=1}^{n-l} x_i}{(n-l)} \ (l = 1, 2, \cdots, m)$$

$${\nu_x}^{(l)} = \frac{\sum_{i=1}^{n-l} x_{i+l}}{(n-l)} \ (l = 1, 2, \cdots, m)$$

$${\mu_y}^{(l)} = \frac{\sum_{i=1}^{n-l} y_i}{(n-l)} \ (l = 1, 2, \cdots, m)$$

$${\nu_y}^{(l)} = \frac{\sum_{i=1}^{n-l} y_{i+l}}{(n-l)} \ (l = 1, 2, \cdots, m)$$

(2) **Usage**

Double precision:

```
ierr = ASL_dfcrcs  (x, n, y, m, rx, ry);
```

Single precision:

```
ierr = ASL_rfcrcs  (x, n, y, m, rx, ry);
```

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	x	{D∗ / R∗}	n	Input	Time series data x_i
2	n	I	1	Input	Number of time series data observed values n
3	y	{D∗ / R∗}	n	Input	Time series data y_i
4	m	I	1	Input	Number of cross correlation coefficients to be obtained m
5	rx	{D∗ / R∗}	m	Output	Cross correlation coefficient ${r_{xy}}^{(l)}$
6	ry	{D∗ / R∗}	m	Output	Cross correlation coefficient ${r_{yx}}^{(l)}$
7	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) n $\geq$ 2

(b) 1 $\leq$ m $\leq$ n

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
1000	For some l, the correlation coefficient ${r_{xy}}^{(l)}$ or ${r_{yx}}^{(l)}$ was not obtained.	0.0 is set for the value of the correlation coefficient that was not obtained.
3000	Restriction (a) was not satisfied.	Processing is aborted.
3100	Restriction (b) was not satisfied.	

(6) **Notes**

None

5.4 SMOOTHING AND DEMAND FORECASTING

5.4.1 ASL_dfasma, ASL_rfasma
Moving Averages

(1) **Function**

Using n given time series data values represented as follows:

$$x_1, x_2, \cdots, x_n$$

and m specified weights represented as follows:

$$w_1, w_2, \cdots, w_m$$

the ASL_dfasma or ASL_rfasma obtains the weighted moving average M_k^w. The weighted moving average M_k^w is defined as follows.

$$M_k^w = \sum_{j=1}^{m} \frac{(x_{k+j-1} \cdot w_j)}{\sum_{j=1}^{m} w_j} \ (k = 1, 2, \cdots, n - m + 1)$$

m can be called the smoothing bandwidth.

(2) **Usage**

Double precision:

```
ierr = ASL_dfasma  (a, n, m, wa, av, isw);
```

Single precision:

```
ierr = ASL_rfasma  (a, n, m, wa, av, isw);
```

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	a	{D∗ / R∗}	n	Input	Time series data x_i
2	n	I	1	Input	Number of time series data n
3	m	I	1	Input	Smoothing bandwidth m
4	wa	{D∗ / R∗}	m	Input	Weight w_j
5	av	{D∗ / R∗}	n − m + 1	Output	Moving average M_k^w
6	isw	I	1	Input	Weight specification switch isw=0: Input weight w_j isw=1: Set all weights to 1.0
7	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) n ≥ 2

(b) 0 < m ≤ n

(c) wa[0] + ··· + wa[m − 1] > 0

(d) isw ∈ {0, 1}

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
1000	Restriction (d) was not satisfied.	All weights are set to 1.0 and processing continues.
3000	Restriction (a) was not satisfied.	Processing is aborted.
3010	Restriction (b) was not satisfied.	
3020	Restriction (c) was not satisfied.	

(6) **Notes**

None

(7) **Example**

(a) Problem

Obtain the moving averages for the following time series data:

a[0] =	71.8	a[12] =	62.3
a[1] =	73.2	a[13] =	51.2
a[2] =	63.8	a[14] =	48.2
a[3] =	60.0	a[15] =	29.8
a[4] =	58.3	a[16] =	34.3
a[5] =	57.2	a[17] =	24.0
a[6] =	48.5	a[18] =	22.9
a[7] =	53.5	a[19] =	31.8
a[8] =	69.3	a[20] =	70.0
a[9] =	68.7	a[21] =	106.7
a[10] =	73.4	a[22] =	138.5
a[11] =	74.9	a[23] =	146.1

to which the following weights are assigned:

wa[0] =	−3.0
wa[1] =	12.0
wa[2] =	17.0
wa[3] =	12.0
wa[4] =	−3.0

(b) Input data

Array a and wa, n=24, m=5 and isw=0.

(c) Main program

```
/*      C interface example for ASL_dfasma */

#include <stdio.h>
#include <stdlib.h>
#include <asl.h>

int main()
{
    double *a;
    int n, m;
    double *wa, *av;
    int isw;
    int ierr;

    int i;

    FILE *fp;


    n=24;
    m=5;
    isw=0;

    fp = fopen( "dfasma.dat", "r" );
    if( fp == NULL )
    {
        printf( "file open error\n" );
        return -1;
    }

    printf( "    *** ASL_dfasma ***\n" );
    printf( "\n     ** Input **\n\n" );
    printf( "\t  n=%3d\n", n );
    printf( "\t  m=%3d\n", m );
    printf( "\tisw=%3d\n\n", isw );

    a = ( double * )malloc((size_t)( sizeof(double) * (n) ));
    if( a == NULL )
    {
        printf( "no enough memory for array a\n" );
        return -1;
    }

    wa = ( double * )malloc((size_t)( sizeof(double) * (m) ));
    if( wa == NULL )
    {
        printf( "no enough memory for array wa\n" );
        return -1;
    }

    av = ( double * )malloc((size_t)( sizeof(double) * (n-m+1) ));
```

```
    if( av == NULL )
    {
        printf( "no enough memory for array av\n" );
        return -1;
    }

    printf( "\tTime series data(Array a)" );
    for( i=0 ; i<n ; i++ )
    {
        fscanf( fp, "%lf", &a[i] );
        if( i%5 == 0 )
        {
            printf( "\n\t" );
        }
        printf( "%8.3g", a[i] );
    }
    printf( "\n\n" );

    printf( "\tWeight(Array wa)\n" );
    printf( "\t" );
    for( i=0 ; i<m ; i++ )
    {
        fscanf( fp, "%lf", &wa[i] );
        printf( "%8.3g", wa[i] );
    }
    printf( "\n" );

    fclose( fp );

    ierr = ASL_dfasma(a, n, m, wa, av, isw);

    printf( "\n     ** Output **\n\n" );
    printf( "\tierr = %6d\n\n", ierr );

    printf( "\tMoving averages(Array av)" );
    for( i=0 ; i<n-m+1 ; i++ )
    {
        if( i%5 == 0 )
        {
            printf( "\n\t" );
        }
        printf( "%8.3g", av[i] );
    }
    printf( "\n" );

    free( a );
    free( wa );
    free( av );

    return 0;
}
```

(d) Output results

```
*** ASL_dfasma ***

 ** Input **

  n= 24
  m=  5
isw=  0

Time series data(Array a)
    71.8    73.2    63.8      60    58.3
    57.2    48.5    53.5    69.3    68.7
    73.4    74.9    62.3    51.2    48.2
    29.8    34.3      24    22.9    31.8
      70     107     139     146

Weight(Array wa)
      -3      12      17      12      -3

 ** Output **

ierr =      0

Moving averages(Array av)
    65.5    59.8    58.9    54.7    50.6
    55.6    65.1    71.3    73.6    72.6
    63.1    53.8    42.9    36.3      29
      26    21.3    36.1    67.7     108
```

5.4.2 ASL_dfdpes, ASL_rfdpes Single Exponential Smoothing

(1) **Function**

The single exponential smoothing equation for a given time series $\cdots, x_{n-1}, x_n$ (x_n is the most recent data) is defined as follows.

$$S_t = \alpha x_t + (1-\alpha) S_{t-1} \ (t = M, M+1, \cdots, n; M \to -\infty)$$

Here, S_t is the smoothing value of x_t and α is the smoothing constant. Now, assume the forecasting model has the following kind of structure.

$$x_t = a + \varepsilon_t$$

Here, a is a constant, and ε_t is an error term that independently obeys $N(0, \sigma^2)$. At this time, the mathematical expectation value E_t at time t and the forecast value E_{t+L} at L periods after t are as follows.

$$E_{t+L} = E_t = S_t$$

The ASL_dfdpes or ASL_rfdpes obtains the smoothing value S_j related to the following given time series data:

$$x_1, x_2, \cdots, x_n$$

(where, x_n is the most recent data). The definition of S_j is as follows.

$$S_1 = \frac{1}{m} \sum_{i=1}^{m} x_i \text{ (initial value)}$$

$$S_j = \alpha x_{j+m-1} + (1-\alpha) S_{j-1} \ (j = 2, 3, \cdots, n-m+1)$$

(2) **Usage**

Double precision:

```
ierr = ASL_dfdpes  (a, n, alh, in, ev);
```

Single precision:

```
ierr = ASL_rfdpes  (a, n, alh, in, ev);
```

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	a	{D∗ / R∗}	n	Input	Time series data x_j
2	n	I	1	Input	Number of time series data n
3	alh	{D / R}	1	Input	Smoothing constant α
4	in	I	1	Input	Average number of terms for the initial value setting m Default value: 2 (when in=0)
5	ev	{D∗ / R∗}	n − in + 1	Output	Smoothing value S_j
6	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) n > 0

(b) 0 ≤ in ≤ n

(c) 0.0 < alh < 1.0

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
1000	n = in (when n ≠ 0)	Only ev[0] is calculated.
3000	Restriction (a) was not satisfied.	Processing is aborted.
3100	Restriction (b) was not satisfied.	
3200	Restriction (c) was not satisfied.	

(6) **Notes**

None

5.4.3 ASL_dfdped, ASL_rfdped Double Exponential Smoothing

(1) **Function**

The double exponential smoothing equation for a given time series $\cdots, x_{n-1}, x_n$ (x_n is the most recent data) is defined as follows.

$$S_t = \alpha x_t + (1-\alpha) S_{t-1}$$

$$D_t = \alpha S_t + (1-\alpha) D_{t-1} \ (t = M, M+1, \cdots, n; M \to -\infty)$$

Here, S_t is the single exponential smoothing value of x_t, D_t is the double exponential smoothing value of x_t, and α is the smoothing constant. Now, assume the forecasting model has the following kind of structure.

$$x_t = a + bt + \varepsilon_t$$

Here, a and b are constants, and ε_t is an error term that independently obeys $N(0, \sigma^2)$. At this time, the mathematical expectation value E_t at time t and the forecast value E_{t+L} at L periods after t are as follows.

$$E_t = 2S_t - D_t$$

$$E_{t+L} = (2S_t - D_t) + B_t \cdot L$$

Here, B_t, which represents the linear trend estimate, is defined by the following equation.

$$B_t = \frac{\alpha}{1-\alpha}(S_t - D_t)$$

The ASL_dfdped or ASL_rfdped obtains the smoothing value E_k, forecast value E_{k+L}, and linear trend estimate B_k related to the following given time series data:

$$x_1, x_2, \cdots, x_n$$

(where, x_n is the most recent data). The definitions of these quantities are as follows.

$$S_1 = \frac{1}{m}\sum_{i=1}^{m} x_i \text{ (initial value)}$$

$$D_1 = S_1 \text{ (initial value)}$$

$$S_j = \alpha x_{j+m-1} + (1-\alpha) S_{j-1}$$

$$D_j = \alpha S_j + (1-\alpha) D_{j-1} \ (j = 2, 3, \cdots, n-m+1)$$

$$E_k = 2S_k - D_k$$

$$B_k = \frac{\alpha}{1-\alpha}(S_k - D_k)$$

$$E_{k+L} = E_k + B_k L \ (k = 1, 2, \cdots, n-m+1)$$

(2) **Usage**

Double precision:

ierr = ASL_dfdped (a, n, alh, in, m, ev, av, tr);

Single precision:

ierr = ASL_rfdped (a, n, alh, in, m, ev, av, tr);

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	a	{D∗ / R∗}	n	Input	Time series data x_j
2	n	I	1	Input	Number of time series data n
3	alh	{D / R}	1	Input	Smoothing constant α
4	in	I	1	Input	Average number of terms for the initial value setting m Default value: 2 (when in=0)
5	m	I	1	Input	Time lag L
6	ev	{D∗ / R∗}	n − in + 1	Output	Smoothing value E_k
7	av	{D∗ / R∗}	n − in + 1	Output	Forecast value E_{k+L}
8	tr	{D∗ / R∗}	n − in + 1	Output	Linear trend estimate B_k
9	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) n > 0

(b) 0 ≤ in ≤ n

(c) 0.0 < alh < 1.0

(d) m ≥ 0

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
1000	n = in	ev[0] and av[0] are calculated, and 0.0 is set for tr[0].
3000	Restriction (a) was not satisfied.	Processing is aborted.
3100	Restriction (b) was not satisfied.	
3200	Restriction (c) was not satisfied.	
3300	Restriction (d) was not satisfied.	

(6) **Notes**

(a) When a model is forecasted having this structure pattern, at least two data values are required. A forecast performed using data for one point is statistically meaningless.

5.4.4 ASL_dfdpet, ASL_rfdpet Triple Exponential Smoothing

(1) **Function**

The triple exponential smoothing equation for a given time series $\cdots, x_{n-1}, x_n$ (x_n is the most recent data) is defined as follows.

$$S_t = \alpha x_t + (1-\alpha)S_{t-1}$$

$$D_t = \alpha S_t + (1-\alpha)D_{t-1}$$

$$T_t = \alpha D_t + (1-\alpha)T_{t-1} \ (t = M, M+1, \cdots, n; M \to -\infty)$$

Here, S_t is the single exponential smoothing value of x_t, D_t is the double exponential smoothing value of x_t, T_t is the triple exponential smoothing value of x_t, and α is the smoothing constant. Now, assume the forecasting model has the following kind of structure.

$$x_t = a + bt + \frac{c}{2}t^2 + \varepsilon_t$$

Here, a, b and c are constants, and ε_t is an error term that independently obeys $N(0, \sigma^2)$. At this time, the mathematical expectation value E_t at time t and the forecast value E_{t+L} at L periods after t are as follows.

$$E_t = 3S_t - 3D_t + T_t$$

$$E_{t+L} = E_t + B_t L + \frac{C_t}{2}L^2$$

Here, B_t, which represents the linear trend estimate, and C_t, which represents the quadratic trend estimate, are defined by the following equations.

$$B_t = \frac{\alpha}{2(1-\alpha)^2}\{(6-5\alpha)S_t - 2(5-4\alpha)D_t + (4-3\alpha)T_t\}$$

$$C_t = \frac{(\alpha)^2}{(1-\alpha)^2}(S_t - 2D_t + T_t)$$

The ASL_dfdpet or ASL_rfdpet obtains the smoothing value E_k, forecast value E_{k+L}, linear trend estimate B_k, and quadratic trend estimate C_k related to the following given time series data:

$$x_1, x_2, \cdots, x_n$$

(where, x_n is the most recent data). The definitions of these quantities are as follows.

$$S_1 = \frac{1}{m}\sum_{i=1}^{m} x_i \text{ (initial value)}$$

$$D_1 = S_1 = T_1 \text{ (initial value)}$$

$$S_j = \alpha x_{j+m-1} + (1-\alpha)S_{j-1}$$

$$D_j = \alpha S_j + (1-\alpha)D_{j-1}$$

$$T_j = \alpha D_j + (1-\alpha)T_{j-1} \ (j = 2, 3, \cdots, n-m+1)$$

$$E_k = 3S_k - 3D_k + T_k$$

$$B_k = \frac{\alpha}{2(1-\alpha)^2}\{(6-5\alpha)S_k - 2(5-4\alpha)D_k + (4-3\alpha)T_k\}$$

$$C_k = \frac{(\alpha)^2}{(1-\alpha)^2}(S_k - 2D_k + T_k)$$

$$E_{k+L} = E_k + B_k L + \frac{C_k}{2}L^2 \ (k = 1, 2, \cdots, n-m+1)$$

(2) **Usage**

Double precision:

ierr = ASL_dfdpet (a, n, alh, in, m, ev, av, tr, qr);

Single precision:

ierr = ASL_rfdpet (a, n, alh, in, m, ev, av, tr, qr);

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	a	{D∗ / R∗}	n	Input	Time series data x_j
2	n	I	1	Input	Number of time series data n
3	alh	{D / R}	1	Input	Smoothing constant α
4	in	I	1	Input	Average number of terms for the initial value setting Default value: 2 (When in=0)
5	m	I	1	Input	Time lag L
6	ev	{D∗ / R∗}	n − in + 1	Output	Smoothing value E_k
7	av	{D∗ / R∗}	n − in + 1	Output	Forecast value E_{k+L}
8	tr	{D∗ / R∗}	n − in + 1	Output	Linear trend estimate B_k
9	qr	{D∗ / R∗}	n − in + 1	Output	Quadratic trend estimate C_k
10	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) $n > 0$

(b) $0 \leq in \leq n$

(c) $0.0 < alh < 1.0$

(d) $m \geq 0$

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
1000	n = in	ev[0] and av[0] are calculated, and 0.0 is set for qr[0] and tr[0].
3000	Restriction (a) was not satisfied.	Processing is aborted.
3100	Restriction (b) was not satisfied.	
3200	Restriction (c) was not satisfied.	
3300	Restriction (d) was not satisfied.	

(6) **Notes**

(a) When a model is forecasted having this structure pattern, at least three data values are required. A forecast performed using data for two or fewer points is statistically meaningless.

Chapter 6

TESTS AND ESTIMATES

6.1 INTRODUCTION

A random sample of size n sampled (using sampling with replacement) from a population is theoretically represented as follows by a set of n random variables having an equal probability distribution since they are mutually independent.

$$S = (X_1, X_2, \cdots, X_n)$$

S is called a sample of size n, and the common probability distribution of the X_i is called the population distribution. The constants that define the population distribution are called parameters, and a parameter for which the value is unknown is called an unknown parameter. Statistical analysis deals with estimates of the values of these kinds of unknown parameters (**statistical estimates**) and decisions on whether or not hypotheses related to the values of unknown parameters can be established (**statistical hypothesis testing**). This library provides functions for calculating the following kinds of estimates and tests.

- Interval estimations
  - Interval estimation of the population ratio according to one set of samples
  - Interval estimation of the population mean according to one set of samples
  - Interval estimation of the difference of the population means according to two sets of independent samples
  - Interval estimation of the population variance due to one set of samples
  - Interval estimation of the population correlation coefficient according to one set of samples
  - Interval estimation of the difference of the population correlation coefficient according to two sets of independent samples
  - Interval estimation in the simple linear regression
- Tests
  - Test of the population ratio according to one set of samples
  - Test of the difference of the population ratios according to two sets of independent samples
  - Test of the population mean according to one set of samples
  - Test of the difference of the population means according to two sets of independent samples
  - Test of the population variance due to one set of samples
  - Test of the population correlation coefficient according to one set of samples
  - Test of the difference of the population correlation coefficients according to two sets of independent samples
  - Test in the simple linear regression

6.1.1 Explanation

(1) **Interval estimation of the population ratio according to one set of samples**
When the number of data having the observed characteristic in the one set of sample data of size n is m, obtain the confidence interval of the population ratio when the confidence level $1-\alpha$ is specified. The confidence interval (p_1, p_2) is defined as follows.

$$p_1 = \frac{m}{(n-m+1)F_1 + m}$$

$$p_2 = \frac{(m+1)F_2}{(m+1)F_2 + (n-m)}$$

Here,

$$\frac{\alpha}{2} = 1 - P(F_1|2(n-m+1), 2m) = 1 - P(F_2|2(m+1), 2(n-m))$$

$P(F|n_1, n_2)$ is the cumulative distribution function (c.d.f.) of a F distribution having numbers of degrees of freedom n_1 and n_2. If the sample ratio $\hat{p}$ is given as $\hat{p} = \frac{m}{n}$,

- when $\hat{p} = 0$:
  the upper bound of one-sided confidence interval in the confidence level $1-\alpha$ is given as follows

  $$p_2 = \frac{F_\alpha}{n + F_\alpha}$$

  where

  $$\alpha = 1 - P(F_\alpha|2, 2n)$$

  $P(F|n_1, n_2)$ is the cumulative distribution function (c.d.f.) of a F distribution having numbers of degrees of freedom n_1 and n_2.
- when $\hat{p} = 1$:
  the lower bound of one-sided confidence interval in the confidence level $1-\alpha$ is given as follows

  $$p_1 = \frac{n}{n + F_\alpha}$$

  where

  $$\alpha = 1 - P(F_\alpha|2, 2n)$$

(2) **Interval estimation of the population mean according to one set of samples**
From the mean μ and variance (or population variance) σ^2 of one set of sample data of size n, obtain the confidence interval of the population mean when the confidence level $1-\alpha$ is specified. The confidence interval (t_1, t_2) is defined as follows.

$$t_1 = \mu - z_{\frac{\alpha}{2}}\sqrt{\beta}$$

$$t_2 = \mu + z_{\frac{\alpha}{2}}\sqrt{\beta}$$

(a) When the population variance is known

$$\frac{\alpha}{2} = 1 - P(z_{\frac{\alpha}{2}})$$

Here, $P(x)$ is the cumulative distribution function (c.d.f.) of a standard normal distribution.

$$\beta = \frac{\sigma^2}{n}$$

σ^2: Population variance

(b) When the population variance is unknown

$$\frac{\alpha}{2} = 1 - P(z_{\frac{\alpha}{2}}, n-1)$$

Here, $P(t, n)$ is the cumulative distribution function (c.d.f.) of a t distribution with n degrees of freedom.

$$\beta = \frac{\sigma^2}{n}$$

σ^2 is an unbiased estimate of the population variance.

(3) **Interval estimation of the difference of the population means according to two sets of independent samples**

From the means μ_1 and μ_2 and variances (or population variances) ${\sigma_1}^2$ and ${\sigma_2}^2$ of two sets of independent sample data of sizes n_1 and n_2, respectively, obtain the confidence interval of the difference of the population means when the confidence level $1-\alpha$ is specified. The confidence interval (t_1, t_2) is defined as follows.

$$t_1 = (\mu_1 - \mu_2) - z_{\frac{\alpha}{2}}\sqrt{\beta_1 + \beta_2}$$

$$t_2 = (\mu_1 - \mu_2) + z_{\frac{\alpha}{2}}\sqrt{\beta_1 + \beta_2}$$

(a) When the population variances are known

$$\frac{\alpha}{2} = 1 - P(z_{\frac{\alpha}{2}})$$

Here, $P(x)$ is the cumulative distribution function (c.d.f.) of a standard normal distribution.

$$\beta_1 = \frac{{\sigma_1}^2}{n_1}, \beta_2 = \frac{{\sigma_2}^2}{n_2}$$

${\sigma_1}^2$, ${\sigma_2}^2$: Population variances of the two sets

(b) When the population variances of the two sets are equal and that value is unknown

$$\frac{\alpha}{2} = 1 - P(z_{\frac{\alpha}{2}}, n_1 + n_2 - 2)$$

Here, $P(t, n)$ is the cumulative distribution function (c.d.f.) of a t distribution with n degrees of freedom.

$$\beta_1 = \frac{{s_p}^2}{n_1}, \beta_2 = \frac{{s_p}^2}{n_2}$$

${s_p}^2$ is defined as follows.

$${s_p}^2 = \frac{(n_1 - 1){\sigma_1}^2 + (n_2 - 1){\sigma_2}^2}{n_1 + n_2 - 2}$$

${\sigma_1}^2$ and ${\sigma_2}^2$ are unbiased estimates of the population variances.

(c) When the population variances of the two sets are not equal and those values are unknown

$$z_{\frac{\alpha}{2}} = \frac{\beta_1 t^{(1)}_{\frac{\alpha}{2}} + \beta_2 t^{(2)}_{\frac{\alpha}{2}}}{\beta_1 + \beta_2}$$

$$\frac{\alpha}{2} = 1 - P(t^{(1)}_{\frac{\alpha}{2}}, n_1 - 1) = 1 - P(t^{(2)}_{\frac{\alpha}{2}}, n_2 - 1)$$

Here, $P(t, n)$ is the cumulative distribution function (c.d.f.) of a t distribution with n degrees of freedom.

$$\beta_1 = \frac{{\sigma_1}^2}{n_1}, \beta_2 = \frac{{\sigma_2}^2}{n_2}$$

${\sigma_1}^2$ and ${\sigma_2}^2$ are unbiased estimates of the population variances.

(4) **Interval estimation of the population variance due to one set of samples**
From the variance (or population variance) σ^2 of one set of sample data of size n, obtain the confidence interval of the population variance when the confidence level (t_1, t_2) is specified. The confidence interval (t_1, t_2) is defined as follows.

$$t_1 = \frac{\sigma^2(n-1)}{{\chi_1}^2}$$

$$t_2 = \frac{\sigma^2(n-1)}{{\chi_2}^2}$$

σ^2 is an unbiased estimate of the population variance. Also,

$$\frac{\alpha}{2} = P({\chi_1}^2; n-1) = 1 - P({\chi_2}^2; n-1)$$

Here, $P(x, y)$ is the cumulative distribution function (c.d.f.) of a χ^2 distribution with y degrees of freedom.

(5) **Interval estimation of the population correlation coefficient according to one set of samples**
From the sample correlation coefficient r of one set of sample data of size n, obtain the confidence interval of the population correlation coefficient ρ when the confidence level $1-\alpha$ is specified. The confidence interval (t_1, t_2) is defined as follows.

$$t_1 = \frac{e^{a-b} - 1}{e^{a-b} + 1}$$

$$t_2 = \frac{e^{a+b} - 1}{e^{a+b} + 1}$$

Here,

$$a = \log_e \frac{1+r}{1-r}$$

$$b = \frac{2z_{\frac{\alpha}{2}}}{\sqrt{n-3}}$$

$$\frac{\alpha}{2} = 1 - P(z_{\frac{\alpha}{2}})$$

$P(z)$ is the cumulative distribution function (c.d.f.) of a standard normal distribution.

(6) **Interval estimation of the difference of the population correlation coefficient according to two sets of independent samples**
From the correlation coefficients r_1 and r_2 of two sets of independent sample data of sizes n_1 and n_2, respectively, obtain the confidence interval of the difference of the population correlation coefficients ρ_1 and ρ_2 when the confidence level $1-\alpha$ is specified. If $\rho_1 = \rho_2 = \rho$, the confidence level of ρ is obtained.

(a) When $\rho_1 = \rho_2 = \rho$
The confidence interval (t_1, t_2) of ρ is defined as follows.

$$t_1 = \frac{e^{a-b} - 1}{e^{a-b} + 1}$$

$$t_2 = \frac{e^{a+b} - 1}{e^{a+b} + 1}$$

Here,

$$a = \frac{2(n_1 - 3)z_1 + 2(n_2 - 3)z_2}{n_1 + n_2 - 6}$$

$$b = \frac{2z_{\frac{\alpha}{2}}}{\sqrt{n_1 + n_2 - 6}}$$

$$z_1 = \frac{1}{2}\log_e \frac{1 + r_1}{1 - r_1}$$

$$z_2 = \frac{1}{2}\log_e \frac{1 + r_2}{1 - r_2}$$

$$\frac{\alpha}{2} = 1 - P(z_{\frac{\alpha}{2}})$$

$P(z)$ is the cumulative distribution function (c.d.f.) of a standard normal distribution.

(b) When $\rho_1 \neq \rho_2$

The confidence interval (t_1, t_2) of $\rho_1 - \rho_2$ is defined as follows.

$$t_1 = \frac{e^{a-b} - 1}{e^{a-b} + 1}$$

$$t_2 = \frac{e^{a+b} - 1}{e^{a+b} + 1}$$

Here,

$$a = \log_e \frac{1 + r_1}{1 - r_1} - \log_e \frac{1 + r_2}{1 - r_2}$$

$$b = 2z_{\frac{\alpha}{2}}\sqrt{\frac{1}{n_1 - 3} + \frac{1}{n_2 - 3}}$$

$$\frac{\alpha}{2} = 1 - P(z_{\frac{\alpha}{2}})$$

$P(z)$ is the cumulative distribution function (c.d.f.) of a standard normal distribution.

(7) **Interval estimation in the simple linear regression**

For regression coefficient a, constant term b, and a given specific data x_0 in the following simple linear regression expression (or regression line) related to one set of sample data $\{x_i, y_i\}$ $(1, \cdots, n)$ of size n

$$\hat{y}_i = ax_i + b$$

obtain the estimate $\hat{y}_0$ and the confidence interval of the confidence level $1 - \alpha$ of the theoretical value $Ax_0 - B$. Assume that y_i corresponding to each x_i is the random sample from the normal population having the mean $Ax_i - B$ and the variance σ^2. Obtain the regression coefficient a and constant term b of the sample data from the following normal equations.

$$\begin{cases} \sum_{i=1}^{n} y_i = a\sum_{i=1}^{n} x_i + bn \\ \sum_{i=1}^{n} x_i y_i = a\sum_{i=1}^{n} x_i^2 + b\sum_{i=1}^{n} x_i \end{cases}$$

The confidence interval (t_1, t_2) is defined as follows.

(a) Regression coefficient

$$t_1 = a - t_{\frac{\alpha}{2}} \cdot s_a$$

$$t_2 = a + t_{\frac{\alpha}{2}} \cdot s_a$$

Here,

$$s_a = \sqrt{\frac{\sigma^2}{\sum (x_i - \mu_x)^2}}$$

i. When the population variance is known

σ^2 : Population variance

$$\frac{\alpha}{2} = 1 - P(t_{\frac{\alpha}{2}})$$

Here, $P(t)$ is the cumulative distribution function (c.d.f.) of a standard normal distribution.

ii. When the population variance is unknown

σ^2 : Unbiased variance of error variation

$$\frac{\alpha}{2} = 1 - P(t_{\frac{\alpha}{2}}|n-2)$$

Here, $P(t|n)$ is the cumulative distribution function (c.d.f.) of a t distribution having number of degrees of freedom n.

(b) Constant term

$$t_1 = a - t_{\frac{\alpha}{2}} \cdot s_b$$

$$t_2 = a + t_{\frac{\alpha}{2}} \cdot s_b$$

Here,

$$s_b = \sqrt{\sigma^2 \left[\frac{1}{n} + \frac{\mu_x^2}{\sum (x_i - \mu_x)^2} \right]}$$

i. When the population variance is known

σ^2 : Population variance

$$\frac{\alpha}{2} = 1 - P(t_{\frac{\alpha}{2}})$$

Here, $P(t)$ is the cumulative distribution function (c.d.f.) of a standard normal distribution.

ii. When the population variance is unknown

σ^2 : Unbiased variance of error variation

$$\frac{\alpha}{2} = 1 - P(t_{\frac{\alpha}{2}}|n-2)$$

Here, $P(t|n)$ is the cumulative distribution function (c.d.f.) of a t distribution having number of degrees of freedom n.

(c) Estimated value

$$t_1 = \hat{y}_0 - t_{\frac{\alpha}{2}} \cdot s_y$$

$$t_2 = \hat{y}_0 + t_{\frac{\alpha}{2}} \cdot s_y$$

Here,

$$s_y = \sqrt{\sigma^2 \left[1 + \frac{1}{n} + \frac{(x_0 - \mu_x)^2}{\sum (x_i - \mu_x)^2} \right]}$$

i. When the population variance is known

σ^2 : Population variance

$$\frac{\alpha}{2} = 1 - P(t_{\frac{\alpha}{2}})$$

Here, $P(t)$ is the cumulative distribution function (c.d.f.) of a standard normal distribution.

ii. When the population variance is unknown

σ^2 : Unbiased variance of error variation

$$\frac{\alpha}{2} = 1 - P(t_{\frac{\alpha}{2}}|n-2)$$

Here, $P(t|n)$ is the cumulative distribution function (c.d.f.) of a t distribution having number of degrees of freedom n.

(d) Theoretical value

$$t_1 = \hat{y}_0 - t_{\frac{\alpha}{2}} \cdot s_0$$

$$t_2 = \hat{y}_0 + t_{\frac{\alpha}{2}} \cdot s_0$$

Here,

$$s_0 = \sqrt{\sigma^2 \left[\frac{1}{n} + \frac{(x_0 - \mu_x)^2}{\sum (x_i - \mu_x)^2} \right]}$$

i. When the population variance is known

σ^2 : Population variance

$$\frac{\alpha}{2} = 1 - P(t_{\frac{\alpha}{2}})$$

Here, $P(t)$ is the cumulative distribution function (c.d.f.) of a standard normal distribution.

ii. When the population variance is unknown

σ^2 : Unbiased variance of error variation

$$\frac{\alpha}{2} = 1 - P(t_{\frac{\alpha}{2}}|n-2)$$

Here, $P(t|n)$ is the cumulative distribution function (c.d.f.) of a t distribution having number of degrees of freedom n.

(8) **Test of the population rate according to one set of samples**

When the number of data having the observed characteristic in the one set of sample data of size n is m, test the hypothesis $p = p_0$ related to the ratio p in the population to which the respective set of sample data belong with the confidence level $1 - \alpha$. The test criteria are as follows.

(a) When the alternative hypothesis is $p \neq p_0$

For f_1 and f_2 defined as follows

$$f_1 = \frac{2(n-m)p_0}{2(m+1)(1-p_0)}$$

$$f_2 = \frac{2m(1-p_0)}{2(n-m+1)p_0}$$

$$\begin{cases} \text{If } f_1 \geq F_1 \text{ or } f_2 \geq F_2, \text{ reject} \\ \text{If } f_1 < F_1 \text{ and } f_2 < F_2, \text{ accept} \end{cases}$$

Here,

$$\frac{\alpha}{2} = 1 - P(F_1|2(n-m+1), 2m) = 1 - P(F_2|2(m+1), 2(n-m))$$

$P(F|n_1, n_2)$ is the cumulative distribution function (c.d.f.) of a F distribution having numbers of degrees of freedom n_1 and n_2.

(b) When the alternative hypothesis is $p < p_0$
For f_1 defined as follows

$$f_1 = \frac{2(n-m)p_0}{2(m+1)(1-p_0)}$$

$$\begin{cases} \text{If } f_1 \geq F_1^*, \text{ reject} \\ \text{If } f_1 < F_1^*, \text{ accept} \end{cases} \text{ Here,}$$

$$\alpha = 1 - P(F_1^* | 2(m+1), 2(n-m))$$

$P(F|n_1, n_2)$ is the cumulative distribution function (c.d.f.) of a F distribution having numbers of degrees of freedom n_1 and n_2.

(c) When the alternative hypothesis is $p > p_0$
For f_2 defined as follows

$$f_2 = \frac{2m(1-p_0)}{2(n-m+1)(1-p_0)}$$

$$\begin{cases} \text{If } f_2 \geq F_2^*, \text{ reject} \\ \text{If } f_2 < F_2^*, \text{ accept} \end{cases} \text{ Here,}$$

$$\alpha = 1 - P(F_2^* | 2(n-m+1), 2m)$$

$P(F|n_1, n_2)$ is the cumulative distribution function (c.d.f.) of a F distribution having numbers of degrees of freedom n_1 and n_2.

(9) **Test of the difference of the population ratios according to two sets of independent samples**
When the number of data having the observed characteristic in the two sets of independent sample data of sizes n_1 and n_2 are m_1 and m_2, respectively, test the hypothesis $p_1 = p_2$ related to the ratios p_1 and p_2 in the population to which the respective sets of sample data belong with the confidence level $1 - \alpha$.
Let the sample ratios in the two sets of independent samples, respectively, be $\hat{p}_1$ and $\hat{p}_2$ as shown below.

$$\hat{p}_1 = \frac{m_1}{n_1}, \hat{p}_2 = \frac{m_2}{n_2}$$

The test criteria are as follows.

(a) When no continuity correction is performed

i. When the alternative hypothesis is $p_1 \neq p_2$
For z defined as follows

$$z = \frac{\hat{p}_1 - \hat{p}_2}{\sqrt{\hat{p}(1-\hat{p})(\frac{1}{n_1} + \frac{1}{n_2})}}$$

$$\begin{cases} \text{If } |z| \geq z_{\frac{\alpha}{2}}, \text{ reject} \\ \text{If } |z| < z_{\frac{\alpha}{2}}, \text{ accept} \end{cases}$$

where, $\hat{p}$ and $\frac{\alpha}{2}$ are as follows.

$$\hat{p} = \frac{n_1\hat{p}_1 + n_2\hat{p}_2}{n_1 + n_2} = \frac{m_1 + m_2}{n_1 + n_2}$$

$$\frac{\alpha}{2} = 1 - P(z_{\frac{\alpha}{2}})$$

Here, $P(z)$ is the c.d.f. value of the standard normal distribution.

ii. When the alternative hypothesis is $p_1 < p_2$

For z defined as follows

$$z = \frac{\hat{p}_1 - \hat{p}_2}{\sqrt{\hat{p}(1-\hat{p})(\frac{1}{n_1} + \frac{1}{n_2})}}$$

$$\begin{cases} \text{If } z \leq -z_\alpha, \text{ reject} \\ \text{If } z > -z_\alpha, \text{ accept} \end{cases}$$

where, $\hat{p}$ and α are as follows.

$$\hat{p} = \frac{n_1\hat{p}_1 + n_2\hat{p}_2}{n_1 + n_2} = \frac{m_1 + m_2}{n_1 + n_2}$$

$$\alpha = 1 - P(z_\alpha)$$

Here, $P(z)$ is the c.d.f. value of the standard normal distribution.

iii. When the alternative hypothesis is $p_1 > p_2$

For z defined as follows

$$z = \frac{\hat{p}_1 - \hat{p}_2}{\sqrt{\hat{p}(1-\hat{p})(\frac{1}{n_1} + \frac{1}{n_2})}}$$

$$\begin{cases} \text{If } z \geq z_\alpha, \text{ reject} \\ \text{If } z < z_\alpha, \text{ accept} \end{cases}$$

where, $\hat{p}$ and α are as follows.

$$\hat{p} = \frac{n_1\hat{p}_1 + n_2\hat{p}_2}{n_1 + n_2} = \frac{m_1 + m_2}{n_1 + n_2}$$

$$\alpha = 1 - P(z_\alpha)$$

Here, $P(z)$ is the c.d.f. value of the standard normal distribution.

(b) When continuity correction is performed

i. When the alternative hypothesis is $p_1 \neq p_2$

For z defined as follows

$$\begin{cases} z = \dfrac{\hat{p}_1 - \hat{p}_2 - 0.5(\frac{1}{n_1} + \frac{1}{n_2})}{\sqrt{\hat{p}(1-\hat{p})(\frac{1}{n_1} + \frac{1}{n_2})}} (\text{when } \hat{p}_1 \geq \hat{p}_2) \\ z = \dfrac{\hat{p}_1 - \hat{p}_2 + 0.5(\frac{1}{n_1} + \frac{1}{n_2})}{\sqrt{\hat{p}(1-\hat{p})(\frac{1}{n_1} + \frac{1}{n_2})}} (\text{when } \hat{p}_1 < \hat{p}_2) \end{cases}$$

$$\begin{cases} \text{If } |z| \geq z_{\frac{\alpha}{2}}, \text{ reject} \\ \text{If } |z| < z_{\frac{\alpha}{2}}, \text{ accept} \end{cases}$$

where, $\hat{p}$ and $\frac{\alpha}{2}$ are as follows.

$$\hat{p} = \frac{n_1\hat{p}_1 + n_2\hat{p}_2}{n_1 + n_2} = \frac{m_1 + m_2}{n_1 + n_2}$$

$$\frac{\alpha}{2} = 1 - P(z_{\frac{\alpha}{2}})$$

Here, $P(z)$ is the c.d.f. value of the standard normal distribution.

ii. When the alternative hypothesis is $p_1 < p_2$

For z defined as follows

$$\begin{cases} z = \dfrac{\hat{p}_1 - \hat{p}_2 - 0.5(\frac{1}{n_1} + \frac{1}{n_2})}{\sqrt{\hat{p}(1-\hat{p})(\frac{1}{n_1} + \frac{1}{n_2})}} (\text{when } \hat{p}_1 \geq \hat{p}_2) \\ z = \dfrac{\hat{p}_1 - \hat{p}_2 + 0.5(\frac{1}{n_1} + \frac{1}{n_2})}{\sqrt{\hat{p}(1-\hat{p})(\frac{1}{n_1} + \frac{1}{n_2})}} (\text{when } \hat{p}_1 < \hat{p}_2) \end{cases}$$

$$\begin{cases} \text{If } z \leq -z_\alpha, \text{ reject} \\ \text{If } z > -z_\alpha, \text{ accept} \end{cases}$$

where, $\hat{p}$ and α are as follows.

$$\hat{p} = \frac{n_1\hat{p}_1 + n_2\hat{p}_2}{n_1 + n_2} = \frac{m_1 + m_2}{n_1 + n_2}$$
$$\alpha = 1 - P(z_\alpha)$$

Here, $P(z)$ is the c.d.f. value of the standard normal distribution.

iii. When the alternative hypothesis is $p_1 > p_2$

For z defined as follows

$$\begin{cases} z = \dfrac{\hat{p}_1 - \hat{p}_2 - 0.5(\frac{1}{n_1} + \frac{1}{n_2})}{\sqrt{\hat{p}(1-\hat{p})(\frac{1}{n_1} + \frac{1}{n_2})}} (\text{when } \hat{p}_1 \geq \hat{p}_2) \\ z = \dfrac{\hat{p}_1 - \hat{p}_2 + 0.5(\frac{1}{n_1} + \frac{1}{n_2})}{\sqrt{\hat{p}(1-\hat{p})(\frac{1}{n_1} + \frac{1}{n_2})}} (\text{when } \hat{p}_1 < \hat{p}_2) \end{cases}$$

$$\begin{cases} \text{If } z \geq z_\alpha, \text{ reject} \\ \text{If } z < z_\alpha, \text{ accept} \end{cases}$$

where, $\hat{p}$ and α are as follows.

$$\hat{p} = \frac{n_1\hat{p}_1 + n_2\hat{p}_2}{n_1 + n_2} = \frac{m_1 + m_2}{n_1 + n_2}$$
$$\alpha = 1 - P(z_\alpha)$$

Here, $P(z)$ is the c.d.f. value of the standard normal distribution.

(10) **Test of the population mean according to one set of samples**

From the mean μ_x and variance (or population variance) σ^2 of one set of sample data of size n, test the hypothesis $\mu = \mu_0$ with the confidence level $1 - \alpha$. The test criteria are as follows.

(a) When the population variance is known

i. When the alternative hypothesis is $\mu \neq \mu_0$

For z defined as follows

$$z = \frac{\mu_x - \mu_0}{\sqrt{\frac{\sigma^2}{n}}}$$

$$\begin{cases} \text{If } |z| \geq z_{\frac{\alpha}{2}}, \text{ reject} \\ \text{If } |z| < z_{\frac{\alpha}{2}}, \text{ accept} \end{cases}$$

where,

σ^2 : Population variance

$$\frac{\alpha}{2} = 1 - P(z_{\frac{\alpha}{2}})$$

$P(z)$ is the cumulative distribution function (c.d.f.) of a standard normal distribution.

ii. When the alternative hypothesis is $\mu < \mu_0$

For z defined as follows

$$z = \frac{\mu_x - \mu_0}{\sqrt{\frac{\sigma^2}{n}}}$$

$$\begin{cases} \text{If } z \leq -z_\alpha, \text{ reject} \\ \text{If } z > -z_\alpha, \text{ accept} \end{cases}$$

where,

σ^2 : Population variance

$$\alpha = 1 - P(z_\alpha)$$

$P(z)$ is the cumulative distribution function (c.d.f.) of a standard normal distribution.

iii. When the alternative hypothesis is $\mu > \mu_0$

For z defined as follows

$$z = \frac{\mu_x - \mu_0}{\sqrt{\frac{\sigma^2}{n}}}$$

$$\begin{cases} \text{If } z \geq z_\alpha, \text{ reject} \\ \text{If } z < z_\alpha, \text{ accept} \end{cases}$$

where,

σ^2 : Population variance

$$\alpha = 1 - P(z_\alpha)$$

$P(z)$ is the cumulative distribution function (c.d.f.) of a standard normal distribution.

(b) When the population variance is unknown

i. When the alternative hypothesis is $\mu \neq \mu_0$

For t defined as follows

$$t = \frac{\mu_x - \mu_0}{\sqrt{\frac{\sigma^2}{n}}}$$

$$\begin{cases} \text{If } |t| \geq t_{\frac{\alpha}{2}}, \text{ reject} \\ \text{If } |t| < t_{\frac{\alpha}{2}}, \text{ accept} \end{cases}$$

where,

σ^2 : Unbiased estimate of population variance

$$\frac{\alpha}{2} = 1 - P(t_{\frac{\alpha}{2}}|n-1)$$

Here, $P(t|n)$ is the cumulative distribution function (c.d.f.) of a t distribution having number of degrees of freedom n.

ii. When the alternative hypothesis is $\mu < \mu_0$

For t defined as follows

$$t = \frac{\mu_x - \mu_0}{\sqrt{\frac{\sigma^2}{n}}}$$

$$\begin{cases} \text{If } t \leq -t_\alpha, \text{ reject} \\ \text{If } t > -t_\alpha, \text{ accept} \end{cases}$$

where,

σ^2 : Unbiased estimate of population variance

$$\alpha = 1 - P(t_\alpha|n-1)$$

Here, $P(t|n)$ is the cumulative distribution function (c.d.f.) of a t distribution having number of degrees of freedom n.

iii. When the alternative hypothesis is $\mu > \mu_0$ For t defined as follows

$$t = \frac{\mu_x - \mu_0}{\sqrt{\frac{\sigma^2}{n}}}$$

$$\begin{cases} \text{If } t \geq t_\alpha, \text{ reject} \\ \text{If } t < t_\alpha, \text{ accept} \end{cases}$$

where,

σ^2 : Unbiased estimate of population variance

$$\alpha = 1 - P(t_\alpha|n-1)$$

Here, $P(t|n)$ is the cumulative distribution function (c.d.f.) of a t distribution having number of degrees of freedom n.

(11) **Test of the difference of the population means according to two sets of independent samples**
From the means μ_{x_1} and μ_{x_2} and variances (or population variances) σ_1^2 and σ_2^2 of two sets of independent sample data of sizes n_1 and n_2, respectively, test the hypothesis $\mu_1 = \mu_2$ related to the means μ_1 and μ_2 in the population to which the respective sets of sample data belong with the confidence level $1 - \alpha$. The test criteria are as follows.

(a) When the population variances are known

i. When the alternative hypothesis is $\mu_1 \neq \mu_2$ For z defined as follows

$$z = \frac{\mu_{x_1} - \mu_{x_2}}{\sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}}$$

$$\begin{cases} \text{If } |z| \geq z_{\frac{\alpha}{2}}, \text{ reject} \\ \text{If } |z| < z_{\frac{\alpha}{2}}, \text{ accept} \end{cases}$$

where $\frac{\alpha}{2}$ is as follows.

$$\frac{\alpha}{2} = 1 - P(z_{\frac{\alpha}{2}})$$

Here, $P(z)$ is the c.d.f. value of the standard normal distribution.
σ_1^2, σ_2^2 : Population variances of the two sets

ii. When the alternative hypothesis is $\mu_1 < \mu_2$
For z defined as follows

$$z = \frac{\mu_{x_1} - \mu_{x_2}}{\sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}}$$

$$\begin{cases} \text{If } z \leq -z_{\alpha}, \text{ reject} \\ \text{If } z > -z_{\alpha}, \text{ accept} \end{cases}$$

where α is as follows.

$$\alpha = 1 - P(z_{\alpha})$$

Here, $P(z)$ is the c.d.f. value of the standard normal distribution.
σ_1^2, σ_2^2 : Population variances of the two sets

iii. When the alternative hypothesis is $\mu_1 > \mu_2$
For z defined as follows

$$z = \frac{\mu_{x_1} - \mu_{x_2}}{\sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}}$$

$$\begin{cases} \text{If } z \geq z_{\alpha}, \text{ reject} \\ \text{If } z < z_{\alpha}, \text{ accept} \end{cases}$$

where α is as follows.

$$\alpha = 1 - P(z_{\alpha})$$

Here, $P(z)$ is the c.d.f. value of the standard normal distribution.
σ_1^2, σ_2^2 : Population variances of the two sets

(b) When the population variances of the two sets are equal and that value is unknown

i. When the alternative hypothesis is $\mu_1 \neq \mu_2$
For t defined as follows

$$t = \frac{\mu_{x_1} - \mu_{x_2}}{\sqrt{s_p^2(\frac{1}{n_1} + \frac{1}{n_2})}}$$

$$\begin{cases} \text{If } |t| \geq t_{\frac{\alpha}{2}}, \text{ reject} \\ \text{If } |t| < t_{\frac{\alpha}{2}}, \text{ accept} \end{cases}$$

where s_p^2 and $\frac{\alpha}{2}$ are as follows.

$$s_p^2 = \frac{(n_1-1)\sigma_1^2 + (n_2-1)\sigma_2^2}{n_1+n_2-2}$$

$$\frac{\alpha}{2} = 1 - P(t_{\frac{\alpha}{2}}|n_1+n_2-2)$$

Here, $P(t|n)$ is the cumulative distribution function (c.d.f.) of a t distribution with n degrees of freedom. σ_1^2, σ_2^2 : Unbiased estimates of the population variances.

ii. When the alternative hypothesis is $\mu_1 < \mu_2$

For t defined as follows

$$t = \frac{\mu_{x_1} - \mu_{x_2}}{\sqrt{s_p^2(\frac{1}{n_1}+\frac{1}{n_2})}}$$

$$\begin{cases} \text{If } t \leq -t_\alpha, \text{ reject} \\ \text{If } t > -t_\alpha, \text{ accept} \end{cases}$$

where s_p^2 and α are as follows.

$$s_p^2 = \frac{(n_1-1)\sigma_1^2 + (n_2-1)\sigma_2^2}{n_1+n_2-2}$$

$$\alpha = 1 - P(t_\alpha|n_1+n_2-2)$$

Here, $P(t|n)$ is the cumulative distribution function (c.d.f.) of a t distribution with n degrees of freedom.

σ_1^2, σ_2^2 : Unbiased estimates of the population variances.

iii. When the alternative hypothesis is $\mu_1 > \mu_2$

For t defined as follows

$$t = \frac{\mu_{x_1} - \mu_{x_2}}{\sqrt{s_p^2(\frac{1}{n_1}+\frac{1}{n_2})}}$$

$$\begin{cases} \text{If } t \geq t_\alpha, \text{ reject} \\ \text{If } t < t_\alpha, \text{ accept} \end{cases}$$

where s_p^2 and α are as follows.

$$s_p^2 = \frac{(n_1-1)\sigma_1^2 + (n_2-1)\sigma_2^2}{n_1+n_2-2}$$

$$\alpha = 1 - P(t_\alpha|n_1+n_2-2)$$

Here, $P(t|n)$ is the cumulative distribution function (c.d.f.) of a t distribution with n degrees of freedom.

σ_1^2, σ_2^2 : Unbiased estimates of the population variances.

(c) When the population variances of the two sets are not equal and those values are unknown

i. When the alternative hypothesis is $\mu_1 \neq \mu_2$

For t defined as follows

$$t = \frac{\mu_{x_1} - \mu_{x_2}}{\sqrt{\frac{\sigma_1^2}{n_1}+\frac{\sigma_2^2}{n_2}}}$$

$$t^*_{\frac{\alpha}{2}} = \frac{\beta_1 t^{(1)}_{\frac{\alpha}{2}} + \beta_2 t^{(2)}_{\frac{\alpha}{2}}}{\beta_1+\beta_2}$$

$$\begin{cases} \text{If } |t| \geq t^*_{\frac{\alpha}{2}}, \text{ reject} \\ \text{If } |t| < t^*_{\frac{\alpha}{2}}, \text{ accept} \end{cases}$$

where $\frac{\alpha}{2}$ is as follows.

$$\frac{\alpha}{2} = 1 - P(t^{(1)}_{\frac{\alpha}{2}}|n_1-1) = 1 - P(t^{(2)}_{\frac{\alpha}{2}}|n_2-1)$$

Here, $P(t|n)$ is the cumulative distribution function (c.d.f.) of a t distribution with n degrees of freedom.

$$\beta_1 = \frac{\sigma_1^2}{n_1}, \beta_2 = \frac{\sigma_2^2}{n_2}$$

σ_1^2, σ_2^2 : Unbiased estimates of the population variances

ii. When the alternative hypothesis is $\mu_1 < \mu_2$

For t defined as follows

$$t = \frac{\mu_{x_1} - \mu_{x_2}}{\sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}}$$

$$t_\alpha^* = \frac{\beta_1 t_\alpha^{(1)} + \beta_2 t_\alpha^{(2)}}{\beta_1 + \beta_2}$$

$$\begin{cases} \text{If } t \leq -t_\alpha^*, \text{ reject} \\ \text{If } t > -t_\alpha^*, \text{ accept} \end{cases}$$

where α is as follows.

$$\alpha = 1 - P(t_\alpha^{(1)}|n_1 - 1) = 1 - P(t_\alpha^{(2)}|n_2 - 1)$$

Here, $P(t|n)$ is the cumulative distribution function (c.d.f.) of a t distribution with n degrees of freedom.

$$\beta_1 = \frac{\sigma_1^2}{n_1}, \beta_2 = \frac{\sigma_2^2}{n_2}$$

σ_1^2, σ_2^2 : Unbiased estimates of the population variances

iii. When the alternative hypothesis is $\mu_1 > \mu_2$

For t defined as follows

$$t = \frac{\mu_{x_1} - \mu_{x_2}}{\sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}}$$

$$t_\alpha^* = \frac{\beta_1 t_\alpha^{(1)} + \beta_2 t_\alpha^{(2)}}{\beta_1 + \beta_2}$$

$$\begin{cases} \text{If } t \geq t_\alpha^*, \text{ reject} \\ \text{If } t < t_\alpha^*, \text{ accept} \end{cases}$$

where α is as follows.

$$\alpha = 1 - P(t_\alpha^{(1)}|n_1 - 1) = 1 - P(t_\alpha^{(2)}|n_2 - 1)$$

Here, $P(t|n)$ is the cumulative distribution function (c.d.f.) of a t distribution with n degrees of freedom.

$$\beta_1 = \frac{\sigma_1^2}{n_1}, \beta_2 = \frac{\sigma_2^2}{n_2}$$

σ_1^2, σ_2^2 : Unbiased estimates of the population variances

(12) **Test of the population variance due to one set of samples**

From the variance (or population variance) s^2 of one set of sample data of size n, test the hypothesis $\sigma^2 = \sigma_0^2$ related to the population variance σ^2 in the population to which the respective set of sample data belong with the confidence level $1 - \alpha$. The test criteria are as follows.

(a) When the alternative hypothesis is $\sigma^2 \neq \sigma_0^2$

For χ^2 defined as follows

$$\chi^2 = \frac{(n-1)s^2}{\sigma_0^2}$$

$\begin{cases} \text{If } \chi^2 \leq \chi^2_{1-\frac{\alpha}{2}} \text{ or } \chi^2 \geq \chi^2_{\frac{\alpha}{2}}, \text{ reject} \\ \text{If } \chi^2_{1-\frac{\alpha}{2}} < \chi^2 < \chi^2_{\frac{\alpha}{2}}, \text{ accept} \end{cases}$
where $\frac{\alpha}{2}$ is as follows.

$$\frac{\alpha}{2} = 1 - P(\chi^2_{\frac{\alpha}{2}}|n-1) = P(\chi^2_{1-\frac{\alpha}{2}}|n-1)$$

$P(\chi^2|n)$ is the cumulative distribution function (c.d.f.) of a χ^2 distribution having number of degrees of freedom n.
s^2 : Unbiased estimate of population variance

(b) When the alternative hypothesis is $\sigma^2 < \sigma_0^2$
For χ^2 defined as follows

$$\chi^2 = \frac{(n-1)s^2}{\sigma_0^2}$$

$\begin{cases} \text{If } \chi^2 \leq \chi^2_{1-\alpha}, \text{ reject} \\ \text{If } \chi^2 > \chi^2_{1-\alpha}, \text{ accept} \end{cases}$
where α is as follows.

$$\alpha = P(\chi^2_{1-\alpha}|n-1)$$

$P(\chi^2|n)$ is the cumulative distribution function (c.d.f.) of a χ^2 distribution having number of degrees of freedom n.
s^2 : Unbiased estimate of population variance

(c) When the alternative hypothesis is $\sigma^2 > \sigma_0^2$
For χ^2 defined as follows

$$\chi^2 = \frac{(n-1)s^2}{\sigma_0^2}$$

$\begin{cases} \text{If } \chi^2 \geq \chi^2_{\alpha}, \text{ reject} \\ \text{If } \chi^2 < \chi^2_{\alpha}, \text{ accept} \end{cases}$
where α is as follows.

$$\alpha = P(\chi^2_{\alpha}|n-1)$$

$P(\chi^2|n)$ is the cumulative distribution function (c.d.f.) of a χ^2 distribution having number of degrees of freedom n.
s^2 : Unbiased estimate of population variance

(13) **Test of the population correlation coefficient according to one set of samples**
From the sample correlation coefficient r of one set of sample data of size n, test the hypothesis $\rho = \rho_0$ related to the population correlation coefficient ρ in the population to which the respective set of sample data belong with the confidence level $1 - \alpha$. The test criteria are as follows.

(a) Hypothesis: $\rho = 0$

i. When the alternative hypothesis is $\rho \neq 0$
For t defined as follows

$$t = r\sqrt{\frac{n-2}{1-r^2}}$$

$\begin{cases} \text{If } |t| \geq t_{\frac{\alpha}{2}}, \text{ reject} \\ \text{If } |t| < t_{\frac{\alpha}{2}}, \text{ accept} \end{cases}$
where, $\frac{\alpha}{2}$ is as follows.

$$\frac{\alpha}{2} = 1 - P(t_{\frac{\alpha}{2}}|n-2)$$

Here, $P(t|n)$ is the cumulative distribution function (c.d.f.) of a t distribution with n degrees of freedom.

ii. When the alternative hypothesis is $\rho < 0$

For t defined as follows

$$t = r\sqrt{\frac{n-2}{1-r^2}}$$

$$\begin{cases} \text{If } t \geq -t_\alpha, \text{ reject} \\ \text{If } t < -t_\alpha, \text{ accept} \end{cases}$$

where, $\frac{\alpha}{2}$ is as follows.

$$\frac{\alpha}{2} = 1 - P(t_\alpha|n-2)$$

Here, $P(t|n)$ is the cumulative distribution function (c.d.f.) of a t distribution with n degrees of freedom.

iii. When the alternative hypothesis is $\rho > 0$

For t defined as follows

$$t = r\sqrt{\frac{n-2}{1-r^2}}$$

$$\begin{cases} \text{If } t \geq t_\alpha, \text{ reject} \\ \text{If } t < t_\alpha, \text{ accept} \end{cases}$$

where, $\frac{\alpha}{2}$ is as follows.

$$\frac{\alpha}{2} = 1 - P(t_\alpha|n-2)$$

Here, $P(t|n)$ is the cumulative distribution function (c.d.f.) of a t distribution with n degrees of freedom.

(b) Hypothesis: $\rho = \rho_0$

i. When the alternative hypothesis is $\rho \neq \rho_0$

For t defined as follows

$$t = (z - z_0)\sqrt{n-3}$$

$$\begin{cases} \text{If } |t| \geq z_{\frac{\alpha}{2}}, \text{ reject} \\ \text{If } |t| < z_{\frac{\alpha}{2}}, \text{ accept} \end{cases}$$

where, z, z_0 and $\frac{\alpha}{2}$ are as follows.

$$z = \frac{1}{2}\log_e \frac{1+r}{1-r}$$

$$z_0 = \frac{1}{2}\log_e \frac{1+\rho_0}{1-\rho_0}$$

$$\frac{\alpha}{2} = 1 - P(z_{\frac{\alpha}{2}})$$

Here, $P(z)$ is the c.d.f. value of the standard normal distribution.

ii. When the alternative hypothesis is $\rho < \rho_0$

For t defined as follows

$$t = (z - z_0)\sqrt{n-3}$$

$$\begin{cases} \text{If } t \leq -z_\alpha, \text{ reject} \\ \text{If } t > -z_\alpha, \text{ accept} \end{cases}$$

where, z, z_0 and $\frac{\alpha}{2}$ are as follows.

$$z = \frac{1}{2}\log_e \frac{1+r}{1-r}$$

$$z_0 = \frac{1}{2}\log_e \frac{1+\rho_0}{1-\rho_0}$$

$$\frac{\alpha}{2} = 1 - P(z_\alpha)$$

Here, $P(z)$ is the c.d.f. value of the standard normal distribution.

iii. When the alternative hypothesis is $\rho > \rho_0$

For t defined as follows

$$t = (z - z_0)\sqrt{n-3}$$

$$\begin{cases} \text{If } t \geq z_\alpha, \text{ reject} \\ \text{If } t < z_\alpha, \text{ accept} \end{cases}$$

where, z, z_0 and $\frac{\alpha}{2}$ are as follows.

$$z = \frac{1}{2}\log_e \frac{1+r}{1-r}$$

$$z_0 = \frac{1}{2}\log_e \frac{1+\rho_0}{1-\rho_0}$$

$$\frac{\alpha}{2} = 1 - P(z_\alpha)$$

Here, $P(z)$ is the c.d.f. value of the standard normal distribution.

(14) **Test of the difference of the population correlation coefficients according to two sets of independent samples**

From the correlation coefficients r_1 and r_2 of two sets of independent sample data of sizes n_1 and n_2, respectively, test the hypothesis $\rho_1 = \rho_2$ related to the population correlation coefficients ρ_1 and ρ_2 in the population to which the respective set of sample data belong with the confidence level $1 - \alpha$. The test criteria are as follows.

(a) When the alternative hypothesis is $\rho_1 \neq \rho_2$

For t defined as follows

$$t = \frac{z_1 - z_2}{\sqrt{\frac{1}{n_1-3} + \frac{1}{n_2-3}}}$$

$$\begin{cases} \text{If } |t| \geq z_{\frac{\alpha}{2}}, \text{ reject} \\ \text{If } |t| < z_{\frac{\alpha}{2}}, \text{ accept} \end{cases}$$

where, z_1, z_2 and $\frac{\alpha}{2}$ are as follows.

$$z_1 = \frac{1}{2}\log_e \frac{1+r_1}{1-r_1}$$

$$z_2 = \frac{1}{2}\log_e \frac{1+r_2}{1-r_2}$$

$$\frac{\alpha}{2} = 1 - P(z_{\frac{\alpha}{2}})$$

Here, $P(z)$ is the c.d.f. value of the standard normal distribution.

(b) When the alternative hypothesis is $\rho_1 < \rho_2$

For t defined as follows

$$t = \frac{z_1 - z_2}{\sqrt{\frac{1}{n_1-3} + \frac{1}{n_2-3}}}$$

$$\begin{cases} \text{If } t \leq -z_\alpha, \text{ reject} \\ \text{If } t > -z_\alpha, \text{ accept} \end{cases}$$

where, z_1, z_2 and α are as follows.

$$z_1 = \frac{1}{2}\log_e \frac{1+r_1}{1-r_1}$$

$$z_2 = \frac{1}{2}\log_e \frac{1+r_2}{1-r_2}$$

$$\alpha = 1 - P(z_\alpha)$$

Here, $P(z)$ is the c.d.f. value of the standard normal distribution.

(c) When the alternative hypothesis is $\rho_1 > \rho_2$
For t defined as follows

$$t = \frac{z_1 - z_2}{\sqrt{\frac{1}{n_1-3} + \frac{1}{n_2-3}}}$$

$$\begin{cases} \text{If } t \geq z_\alpha, \text{ reject} \\ \text{If } t < z_\alpha, \text{ accept} \end{cases}$$

where, z_1, z_2 and α are as follows.

$$z_1 = \frac{1}{2}\log_e \frac{1+r_1}{1-r_1}$$

$$z_2 = \frac{1}{2}\log_e \frac{1+r_2}{1-r_2}$$

$$\alpha = 1 - P(z_\alpha)$$

Here, $P(z)$ is the c.d.f. value of the standard normal distribution.

(15) **Test in the simple linear regression**
For regression coefficient a and constant term b in the following simple linear regression expression (or regression line) related to one set of sample data $\{x_i, y_i\}$ $(1, \cdots, n)$ of size n

$$\hat{y}_i = ax_i + b$$

test the hypothesis related to the regression coefficient A and constant term B in the population to which the respective set of sample data belong with the confidence level $1 - \alpha$. Assume that y_i corresponding to each x_i is the random sample from the normal population having the mean $Ax_i - B$ and the variance σ^2. Obtain the regression coefficient a and constant term b of the sample data from the following normal equations.

$$\begin{cases} \sum_{i=1}^{n} y_i = a\sum_{i=1}^{n} x_i + bn \\ \sum_{i=1}^{n} x_i y_i = a\sum_{i=1}^{n} x_i^2 + b\sum_{i=1}^{n} x_i \end{cases}$$

The test criteria are as follows.

(a) Regression coefficient
Hypothesis: $A = A_0$

i. When the population variance is known

A. When the alternative hypothesis is $A \neq A_0$
For t defined as follows

$$t = \frac{a - A_0}{s_a}$$

$$\begin{cases} \text{If } |t| \geq z_{\frac{\alpha}{2}}, \text{ reject} \\ \text{If } |t| < z_{\frac{\alpha}{2}}, \text{ accept} \end{cases}$$

where s_a is as follows.

$$s_a = \sqrt{\frac{\sigma^2}{\sum (x_i - \mu_x)^2}}$$

σ^2 : Population variance

$$\frac{\alpha}{2} = 1 - P(z_{\frac{\alpha}{2}})$$

Here, $P(z)$ is the c.d.f. value of the standard normal distribution.

B. When the alternative hypothesis is $A < A_0$

For t defined as follows

$$t = \frac{a - A_0}{s_a}$$

$$\begin{cases} \text{If } t \geq -z_{\alpha}, \text{ reject} \\ \text{If } t < -z_{\alpha}, \text{ accept} \end{cases}$$

where s_a is as follows.

$$s_a = \sqrt{\frac{\sigma^2}{\sum (x_i - \mu_x)^2}}$$

σ^2 : Population variance

$$\alpha = 1 - P(z_{\alpha})$$

Here, $P(z)$ is the c.d.f. value of the standard normal distribution.

C. When the alternative hypothesis is $A > A_0$

For t defined as follows

$$t = \frac{a - A_0}{s_a}$$

$$\begin{cases} \text{If } t \geq z_{\alpha}, \text{ reject} \\ \text{If } t < z_{\alpha}, \text{ accept} \end{cases}$$

where s_a is as follows.

$$s_a = \sqrt{\frac{\sigma^2}{\sum (x_i - \mu_x)^2}}$$

σ^2 : Population variance

$$\alpha = 1 - P(z_{\alpha})$$

Here, $P(z)$ is the c.d.f. value of the standard normal distribution.

ii. When the population variance is unknown

A. When the alternative hypothesis is $A < A_0$

For t defined as follows

$$t = \frac{a - A_0}{s_a}$$

$$\begin{cases} \text{If } |t| \geq t_{\frac{\alpha}{2}}, \text{ reject} \\ \text{If } |t| < t_{\frac{\alpha}{2}}, \text{ accept} \end{cases}$$

where s_a is as follows.

$$s_a = \sqrt{\frac{\sigma^2}{\sum (x_i - \mu_x)^2}}$$

σ^2 : Unbiased variance of error variation

$$\frac{\alpha}{2} = 1 - P(t_{\frac{\alpha}{2}} | n - 2)$$

Here, $P(t|n)$ is the cumulative distribution function (c.d.f.) of a t distribution having number of degrees of freedom n.

B. When the alternative hypothesis is $A < A_0$

For t defined as follows

$$t = \frac{a - A_0}{s_a}$$

$$\begin{cases} \text{If } t \geq -t_\alpha, \text{ reject} \\ \text{If } t < -t_\alpha, \text{ accept} \end{cases}$$

where s_a is as follows.

$$s_a = \sqrt{\frac{\sigma^2}{\sum (x_i - \mu_x)^2}}$$

σ^2 : Unbiased variance of error variation

$$\alpha = 1 - P(t_\alpha | n - 2)$$

Here, $P(t|n)$ is the cumulative distribution function (c.d.f.) of a t distribution having number of degrees of freedom n.

C. When the alternative hypothesis is $A > A_0$

For t defined as follows

$$t = \frac{a - A_0}{s_a}$$

$$\begin{cases} \text{If } t \geq t_\alpha, \text{ reject} \\ \text{If } t < t_\alpha, \text{ accept} \end{cases}$$

where s_a is as follows.

$$s_a = \sqrt{\frac{\sigma^2}{\sum (x_i - \mu_x)^2}}$$

σ^2 : Unbiased variance of error variation

$$\alpha = 1 - P(t_\alpha | n - 2)$$

Here, $P(t|n)$ is the cumulative distribution function (c.d.f.) of a t distribution having number of degrees of freedom n.

(b) Constant term

Hypothesis: $B = B_0$

i. When the population variance is known

A. When the alternative hypothesis is $B \neq B_0$

For t defined as follows

$$t = \frac{b - B_0}{s_b}$$

$$\begin{cases} \text{If } |t| \geq z_{\frac{\alpha}{2}}, \text{ reject} \\ \text{If } |t| < z_{\frac{\alpha}{2}}, \text{ accept} \end{cases}$$

where s_b is as follows.

$$s_b = \sqrt{\sigma^2 \left[\frac{1}{n} + \frac{\mu_x^2}{\sum (x_i - \mu_x)^2} \right]}$$

σ^2 : Population variance

$$\frac{\alpha}{2} = 1 - P(z_{\frac{\alpha}{2}})$$

Here, $P(z)$ is the c.d.f. value of the standard normal distribution.

B. When the alternative hypothesis is $B < B_0$

For t defined as follows

$$t = \frac{b - B_0}{s_b}$$

$$\begin{cases} \text{If } t \geq -z_\alpha, \text{ reject} \\ \text{If } t < -z_\alpha, \text{ accept} \end{cases}$$

where s_b is as follows.

$$s_b = \sqrt{\sigma^2 \left[\frac{1}{n} + \frac{\mu_x^2}{\sum (x_i - \mu_x)^2} \right]}$$

σ^2 : Population variance

$$\alpha = 1 - P(z_\alpha)$$

Here, $P(z)$ is the c.d.f. value of the standard normal distribution.

C. When the alternative hypothesis is $B > B_0$

For t defined as follows

$$t = \frac{b - B_0}{s_b}$$

$$\begin{cases} \text{If } t \geq z_{\frac{\alpha}{2}}, \text{ reject} \\ \text{If } t < z_{\frac{\alpha}{2}}, \text{ accept} \end{cases}$$

where s_b is as follows.

$$s_b = \sqrt{\sigma^2 \left[\frac{1}{n} + \frac{\mu_x^2}{\sum (x_i - \mu_x)^2} \right]}$$

σ^2 : Population variance

$$\alpha = 1 - P(z_\alpha)$$

Here, $P(z)$ is the c.d.f. value of the standard normal distribution.

ii. When the population variance is unknown

A. When the alternative hypothesis is $B \neq B_0$

For t defined as follows

$$t = \frac{b - B_0}{s_b}$$

$$\begin{cases} \text{If } |t| \geq t_{\frac{\alpha}{2}}, \text{ reject} \\ \text{If } |t| < t_{\frac{\alpha}{2}}, \text{ accept} \end{cases}$$

where s_b is as follows.

$$s_b = \sqrt{\sigma^2 \left[\frac{1}{n} + \frac{\mu_x^2}{\sum (x_i - \mu_x)^2} \right]}$$

σ^2 : Unbiased variance of error variation

$$\frac{\alpha}{2} = 1 - P(t_{\frac{\alpha}{2}} | n-2)$$

Here, $P(t|n)$ is the cumulative distribution function (c.d.f.) of a t distribution having number of degrees of freedom n.

B. When the alternative hypothesis is $B < B_0$

For t defined as follows

$$t = \frac{b - B_0}{s_b}$$

$$\begin{cases} \text{If } t \geq -t_\alpha, \text{ reject} \\ \text{If } t < -t_\alpha, \text{ accept} \end{cases}$$

where s_b is as follows.

$$s_b = \sqrt{\sigma^2 \left[\frac{1}{n} + \frac{\mu_x^2}{\sum (x_i - \mu_x)^2} \right]}$$

σ^2 : Unbiased variance of error variation

$$\alpha = 1 - P(t_\alpha | n-2)$$

Here, $P(t|n)$ is the cumulative distribution function (c.d.f.) of a t distribution having number of degrees of freedom n.

C. When the alternative hypothesis is $B > B_0$

For t defined as follows

$$t = \frac{b - B_0}{s_b}$$

$$\begin{cases} \text{If } t \geq t_{\frac{\alpha}{2}}, \text{ reject} \\ \text{If } t < t_{\frac{\alpha}{2}}, \text{ accept} \end{cases}$$

where s_b is as follows.

$$s_b = \sqrt{\sigma^2 \left[\frac{1}{n} + \frac{\mu_x^2}{\sum (x_i - \mu_x)^2} \right]}$$

σ^2 : Unbiased variance of error variation

$$\alpha = 1 - P(t_\alpha | n - 2)$$

Here, $P(t|n)$ is the cumulative distribution function (c.d.f.) of a t distribution having number of degrees of freedom n.

6.1.2 Reference Bibliography

(1) Lehmann, E. L. , "Testing statistical hypotheses", John Wiley and Sons, New York (1959, 2nd ed. 1986)

(2) Snedecar, G. W. , "Statistical methods", Ames, Iowa (1940)

6.2 INTERVAL ESTIMATES

6.2.1 ASL_d3iera, ASL_r3iera

Interval Estimation of the Population Ratio According to One Set of Samples

(1) **Function**

The ASL_d3iera or ASL_r3iera obtains the confidence interval of confidence coefficient $1-\alpha$ related to the population ratio p when the number of samples having the characteristic being observed among one set of sample data of size n is assumed to be m. The confidence interval (p_1, p_2) is defined as follows.

$$p_1 = \frac{m}{(n-m+1)F^{(1)}_{\frac{\alpha}{2}} + m}$$

$$p_2 = \frac{(m+1)F^{(2)}_{\frac{\alpha}{2}}}{(m+1)F^{(2)}_{\frac{\alpha}{2}} + (n-m)}$$

where,

$$\frac{\alpha}{2} = 1 - P(F^{(1)}_{\frac{\alpha}{2}}|2(n-m+1), 2m) = 1 - P(F^{(2)}_{\frac{\alpha}{2}}|2(m+1), 2(n-m))$$

Here, $P(F|n_1, n_2)$ is the value of the cumulative distribution function (c.d.f.) of the F distribution with n_1 and n_2 degrees of freedom. Also, assume the sample ratio $\hat{p} = \frac{m}{n}$ is given as follows.

- When $\hat{p} = 0$

  p_2, which is given as follows:

  $$p_2 = \frac{F(\alpha|2, 2n)}{n + F(\alpha|2, 2n)}$$

  is the upper bound of the one-tailed confidence interval of confidence level $1-\alpha$.

- When $\hat{p} = 1$

  p_1, which is given as follows:

  $$p_1 = \frac{n}{n + F(\alpha|2, 2n)}$$

  is the lower bound of the one-tailed confidence interval of confidence level $1-\alpha$.

(2) **Usage**

Double precision:

ierr = ASL_d3iera (n, m, cl, ci);

Single precision:

ierr = ASL_r3iera (n, m, cl, ci);

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	n	I	1	Input	Number of sample data n
2	m	I	1	Input	Number of samples having the characteristic being observed m
3	cl	{D / R}	1	Input	Confidence level $100(1-\alpha)(\%)$
4	ci	{D∗ / R∗}	2	Output	ci[0]: Confidence interval lower bound p_1 ci[1]: Confidence interval upper bound p_2
5	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) $n > 0$

(b) $m \geq 0$

(c) $n \geq m$

(d) $0.0 \leq cl \leq 100.0$

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
1000	cl=100.0	ci[0]=0.0 and ci[1]=1.0 are performed.
3000	Restriction (a) was not satisfied.	Processing is aborted.
3010	Restriction (b) was not satisfied.	
3020	Restriction (c) was not satisfied.	
3030	Restriction (d) was not satisfied.	

(6) **Notes**

None

(7) **Example**

(a) Problem

When the number of samples having the characteristic being observed among a sample of size 20 is assumed to be 14, obtain the confidence interval of the population ratio with 95% confidence level.

(b) Input data

n=20, m=14 and cl=95.0.

(c) Main program

```
/*      C interface example for ASL_d3iera */

#include <stdio.h>
#include <stdlib.h>
#include <asl.h>

int main()
{
    int n;
    int m;
    double cl;
    double ci[2];
    int ierr;
    int i;


    printf( "    *** ASL_d3iera ***\n" );
    printf( "\n     ** Input **\n\n" );

    n=20;
    m=14;
    cl=90.0e0;

    printf( "\tn   = %6d\n", n);
    printf( "\tm   = %6d\n", m);
    printf( "\tcl  = %8.3g\n", cl);


    ierr = ASL_d3iera(n, m, cl, ci);

    printf( "\n     ** Output **\n\n" );

    printf( "\tierr  = %4d\n", ierr );

    for( i=0 ; i<2 ; i++ )
    {
        printf( "\tci[%2d]= %8.3g\n", i,ci[i] );
    }


    return 0;
}
```

(d) Output results

```
*** ASL_d3iera ***

 ** Input **

n   =     20
m   =     14
cl  =       90

 ** Output **

ierr  =    0
ci[ 0]=    0.492
ci[ 1]=     0.86
```

6.2.2 ASL_d3ieme, ASL_r3ieme
Interval Estimation of the Population Mean According to One Set of Samples

(1) **Function**

The ASL_d3ieme or ASL_r3ieme obtains the confidence interval of the population mean from the mean μ and variance (or population variance) σ^2 of one set of sample data of size n when the confidence level $1-\alpha$ is specified. The confidence interval (t_1, t_2) is defined as follows.

$$t_1 = \mu - z_{\frac{\alpha}{2}}\sqrt{\beta}$$

$$t_2 = \mu + z_{\frac{\alpha}{2}}\sqrt{\beta}$$

(a) When the population variance is known

$$\frac{\alpha}{2} = 1 - P(z_{\frac{\alpha}{2}})$$

Here, $P(x)$ is the cumulative distribution function (c.d.f.) of a standard normal distribution.

$$\beta = \frac{\sigma^2}{n}$$

σ^2: Population variance

(b) When the population variance is unknown

$$\frac{\alpha}{2} = 1 - P(z_{\frac{\alpha}{2}}, n-1)$$

Here, $P(t, n)$ is the cumulative distribution function (c.d.f.) of a t distribution with n degrees of freedom.

$$\beta = \frac{\sigma^2}{n}$$

σ^2 is an unbiased estimate of the population variance.

(2) **Usage**

Double precision:

```
ierr = ASL_d3ieme  (n, xe, xv, cl, ci, isw);
```

Single precision:

```
ierr = ASL_r3ieme  (n, xe, xv, cl, ci, isw);
```

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	n	I	1	Input	Number of sample data n
2	xe	{D / R}	1	Input	Mean of sample data μ
3	xv	{D / R}	1	Input	Variance of sample data or population σ^2
4	cl	{D / R}	1	Input	Confidence level $100(1-\alpha)(\%)$
5	ci	{D∗ / R∗}	2	Output	ci[0]: Confidence interval lower bound t_1 ci[1]: Confidence interval upper bound t_2
6	isw	I	1	Input	isw=1: The variance of the population is entered for xv isw=2: The variance (not an unbiased estimate) of the sample data is entered for xv isw=3: The variance (unbiased estimate) of the sample data is entered for xv
7	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) isw $\in \{1, 2, 3\}$

(b) n ≥ 2

(c) xv > 0.0

(d) $0.0 \leq$ cl ≤ 100.0

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
1000	cl=100.0	The negative minimum value is set for ci[0] and the positive maximum value is set for ci[1].
1010	cl=0.0 (Confidence level is 0.0%.)	The mean is set for ci[0] and ci[1].
3000	Restriction (a) was not satisfied.	Processing is aborted.
3010	Restriction (b) was not satisfied.	
3020	Restriction (c) was not satisfied.	
3030	Restriction (d) was not satisfied.	

(6) **Notes**

None

(7) **Example**

(a) Problem

Obtain the confidence interval of the population mean at a 95% confidence level when the number of sample data is 100, the mean is 42.0, and the unbiased variance is 2.25.

(b) Input data

isw=1, n=100, xe=42.0, xv=2.25 and cl=95.0.

(c) Main program

```
/*      C interface example for ASL_d3ieme */

#include <stdio.h>
#include <asl.h>

int main()
{
    int n;
    double xe;
    double xv;
    double cl;
    double ci[2];
    int isw;
    int ierr;

    isw=1;
    n=100;
    xe=42.0;
    xv=2.25;
    cl=95.0;


    printf( "    *** ASL_d3ieme ***\n" );
    printf( "\n     ** Input **\n\n" );
    printf( "\tisw = %6d\n", isw );
    printf( "\tn = %6d\n", n );
    printf( "\txe = %8.3g\n", xe );
    printf( "\txv = %8.3g\n", xv );
    printf( "\tcl = %8.3g\n", cl );

    ierr = ASL_d3ieme(n, xe, xv, cl, ci, isw);

    printf( "\n     ** Output **\n\n" );
    printf( "\tierr = %6d\n", ierr );

    printf( "\tInterval = (%8.3g, %8.3g)\n", ci[0], ci[1] );

    return 0;
}
```

(d) Output results

```
*** ASL_d3ieme ***

 ** Input **

isw =       1
n =    100
xe =       42
xv =     2.25
cl =       95

 ** Output **

ierr =      0
Interval = (    41.7,     42.3)
```

6.2.3 ASL_d3iesu, ASL_r3iesu
Interval Estimation of the Difference of the Population Means According to Two Sets of Independent Samples

(1) **Function**

The ASL_d3iesu or ASL_r3iesu obtains the confidence interval of the difference of the population means from the means μ_1 and μ_2 and variances (or population variances) ${\sigma_1}^2$ and ${\sigma_2}^2$ of two sets of independent sample data of sizes n_1 and n_2, respectively, when the confidence level $1-\alpha$ is specified. The confidence interval (t_1, t_2) is defined as follows.

$$t_1 = (\mu_1 - \mu_2) - z_{\frac{\alpha}{2}}\sqrt{\beta_1 + \beta_2}$$

$$t_2 = (\mu_1 - \mu_2) + z_{\frac{\alpha}{2}}\sqrt{\beta_1 + \beta_2}$$

(a) When the population variances are known

$$\frac{\alpha}{2} = 1 - P(z_{\frac{\alpha}{2}})$$

Here, $P(x)$ is the cumulative distribution function (c.d.f.) of a standard normal distribution.

$$\beta_1 = \frac{{\sigma_1}^2}{n_1}, \beta_2 = \frac{{\sigma_2}^2}{n_2}$$

${\sigma_1}^2$, ${\sigma_2}^2$: Population variances of the two sets

(b) When the population variances of the two sets are equal and that value is unknown

$$\frac{\alpha}{2} = 1 - P(z_{\frac{\alpha}{2}}, n_1 + n_2 - 2)$$

Here, $P(t, n)$ is the cumulative distribution function (c.d.f.) of a t distribution with n degrees of freedom.

$$\beta_1 = \frac{{s_p}^2}{n_1}, \beta_2 = \frac{{s_p}^2}{n_2}$$

${s_p}^2$ is defined as follows.

$${s_p}^2 = \frac{(n_1 - 1){\sigma_1}^2 + (n_2 - 1){\sigma_2}^2}{n_1 + n_2 - 2}$$

${\sigma_1}^2$ and ${\sigma_2}^2$ are unbiased estimates of the population variances.

(c) When the population variances of the two sets are not equal and those values are unknown

$$z_{\frac{\alpha}{2}} = \frac{\beta_1 t^{(1)}_{\frac{\alpha}{2}} + \beta_2 t^{(2)}_{\frac{\alpha}{2}}}{\beta_1 + \beta_2}$$

$$\frac{\alpha}{2} = 1 - P(t^{(1)}_{\frac{\alpha}{2}}, n_1 - 1) = 1 - P(t^{(2)}_{\frac{\alpha}{2}}, n_2 - 1)$$

Here, $P(t, n)$ is the cumulative distribution function (c.d.f.) of a t distribution with n degrees of freedom.

$$\beta_1 = \frac{{\sigma_1}^2}{n_1}, \beta_2 = \frac{{\sigma_2}^2}{n_2}$$

${\sigma_1}^2$ and ${\sigma_2}^2$ are unbiased estimates of the population variances.

(2) **Usage**

Double precision:

ierr = ASL_d3iesu (n1, xe1, xv1, n2, xe2, xv2, cl, ci, isw);

Single precision:

ierr = ASL_r3iesu (n1, xe1, xv1, n2, xe2, xv2, cl, ci, isw);

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	n1	I	1	Input	Number of sample data n_1
2	xe1	{D / R}	1	Input	Mean of sample data μ_1
3	xv1	{D / R}	1	Input	Variance of sample data or population ${\sigma_1}^2$
4	n2	I	1	Input	Number of sample data n_2
5	xe2	{D / R}	1	Input	Mean of sample data μ_2
6	xv2	{D / R}	1	Input	Variance of sample data or population ${\sigma_2}^2$
7	cl	{D / R}	1	Input	Confidence level $100(1-\alpha)(\%)$
8	ci	{D∗ / R∗}	2	Output	ci[0]: Confidence interval lower bound t_1 ci[1]: Confidence interval upper bound t_2

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
9	isw	I	1	Input	isw=1: The population variances of the two sets are entered for xv1 and xv2 isw=2: The population variances of the two sets are equal, and the variances (not unbiased estimates) of the sample data are entered for xv1 and xv2 isw=3: The population variances of the two sets are equal, and the variances (unbiased estimates) of the sample data are entered for xv1 and xv2 isw=4: The population variances of the two sets are not equal, and the variances (not unbiased estimates) of the sample data are entered for xv1 and xv2 isw=5: The population variances of the two sets are not equal, and the variances (unbiased estimates) of the sample data are entered for xv1 and xv2
10	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) isw $\in \{1, 2, 3, 4, 5\}$

(b) n1, n2 ≥ 2

(c) xv1, xv2 > 0.0

(d) $0.0 \leq$ cl ≤ 100.0

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
1000	cl=100.0	The negative minimum value is set for ci[0] and the positive maximum value is set for ci[1].
3000	Restriction (a) was not satisfied.	Processing is aborted.
3010	Restriction (b) was not satisfied.	
3020	Restriction (c) was not satisfied.	
3030	Restriction (d) was not satisfied.	

(6) **Notes**

None

(7) **Example**

(a) Problem

Obtain the confidence interval of the difference of the population means at a 95% confidence level

from two sets of independent samples for which the sample variances are not equal and the numbers of sample data, the means, and the unbiased variances are 100, 42.0, 2.25 and 50, 30.0, 3.25, respectively.

(b) Input data

isw=5, n1=100, xe1=42.0, xv1=2.25, n2=50, xe2=30.0, xv2=3.25 and cl=95.0.

(c) Main program

```
/*      C interface example for ASL_d3iesu */

#include <stdio.h>
#include <asl.h>

int main()
{
    int n1, n2;
    double xe1, xe2, xv1, xv2;
    double cl;
    double ci[2];
    int isw;
    int ierr;

    isw=   5;
    n1 = 100;
    xe1=42.0;
    xv1=2.25;
    n2 =  50;
    xe2=30.0;
    xv2=3.25;
    cl =95.0;

    printf( "    *** ASL_d3iesu ***\n" );
    printf( "\n     ** Input **\n\n" );
    printf( "\tisw = %6d\n", isw );
    printf( "\tn1  = %6d\n", n1 );
    printf( "\txe1 = %6.3g\n", xe1 );
    printf( "\txv1 = %6.3g\n", xv1 );
    printf( "\tn2  = %6d\n", n2 );
    printf( "\txe2 = %6.3g\n", xe2 );
    printf( "\txv2 = %6.3g\n", xv2 );
    printf( "\tcl  = %6.3g\n", cl );

    ierr = ASL_d3iesu(n1, xe1, xv1, n2, xe2, xv2, cl, ci, isw);

    printf( "\n     ** Output **\n\n" );
    printf( "\tierr = %6d\n", ierr );

    printf( "\tInterval = (%8.3g, %8.3g)\n", ci[0], ci[1] );

    return 0;
}
```

(d) Output results

```
*** ASL_d3iesu ***

 ** Input **

isw =      5
n1  =    100
xe1 =     42
xv1 =   2.25
n2  =     50
xe2 =     30
xv2 =   3.25
cl  =     95

 ** Output **

ierr =      0
Interval = (    11.4,     12.6)
```

6.2.4 ASL_d3ieva, ASL_r3ieva
Interval Estimation of the Population Variance Due to One Set of Samples

(1) **Function**

The ASL_d3ieva or ASL_r3ieva obtains the confidence interval of the population variance from the variance (or population variance) σ^2 of one set of sample data of size n when the confidence level $1-\alpha$ is specified. The confidence interval (t_1, t_2) is defined as follows.

$$t_1 = \frac{\sigma^2(n-1)}{{\chi_1}^2}$$

$$t_2 = \frac{\sigma^2(n-1)}{{\chi_2}^2}$$

σ^2 is an unbiased estimate of the population variance. Also,

$$\frac{\alpha}{2} = P({\chi_1}^2; n-1) = 1 - P({\chi_2}^2; n-1)$$

Here, $P(x, y)$ is the cumulative distribution function (c.d.f.) of a χ^2 distribution with y degrees of freedom.

(2) **Usage**

Double precision:

```
ierr = ASL_d3ieva  (n, xv, cl, ci, isw);
```

Single precision:

```
ierr = ASL_r3ieva  (n, xv, cl, ci, isw);
```

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	n	I	1	Input	Number of sample data n
2	xv	{D / R}	1	Input	Variance of sample data σ^2
3	cl	{D / R}	1	Input	Confidence level $100(1-\alpha)(\%)$
4	ci	{D* / R*}	2	Output	ci[0]: Confidence interval lower bound t_1 ci[1]: Confidence interval upper bound t_2
5	isw	I	1	Input	isw=1: The variance (not an unbiased estimate) of the sample data is entered for xv isw=2: The variance (unbiased estimate) of the sample data is entered for xv
6	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) isw $\in \{1, 2\}$

(b) n ≥ 2

(c) xv > 0.0

(d) $0.0 <$ cl ≤ 100.0

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
1000	cl=100.0	0.0 is set for ci[0] and the positive maximum value is set for ci[1].
3000	Restriction (a) was not satisfied.	Processing is aborted.
3010	Restriction (b) was not satisfied.	
3020	Restriction (c) was not satisfied.	
3030	Restriction (d) was not satisfied.	

(6) **Notes**

(a) When the confidence interval at a $1-\alpha$ confidence level for the variance σ^2 is (t_1, t_2), the confidence interval for the standard deviation is $(\sqrt{t_1}, \sqrt{t_2})$.

(7) **Example**

(a) Problem
Obtain the confidence interval of the population variance at a 95% confidence level when the number of sample data is 25 and the unbiased variance is 29.16.

(b) Input data
isw=2, n=25, xv=29.16 and cl=95.0.

(c) Main program

```
/*      C interface example for ASL_d3ieva */

#include <stdio.h>
#include <asl.h>

int main()
{
    int n;
    double xv;
    double cl;
    double ci[2];
    int isw;
    int ierr;

    isw= 2;
    n  =25;
    xv =29.16;
    cl =95.0;


    printf( "    *** ASL_d3ieva ***\n" );
    printf( "\n     ** Input **\n\n" );
    printf( "\tisw =%6d\n", isw );
    printf( "\tn   =%6d\n", n );
    printf( "\txv  =%6.3g\n", xv );
    printf( "\tcl  =%6.3g\n", cl );

    ierr = ASL_d3ieva(n, xv, cl, ci, isw);

    printf( "\n     ** Output **\n\n" );
    printf( "\tierr = %6d\n", ierr );

    printf( "\tInterval = (%8.3g, %8.3g)\n", ci[0], ci[1] );

    return 0;
}
```

(d) Output results

```
*** ASL_d3ieva ***

 ** Input **

isw =     2
n   =    25
xv  =  29.2
cl  =    95

 ** Output **

ierr =      0
Interval = (    17.8,     56.4)
```

6.2.5 ASL_d3ietc, ASL_r3ietc
Interval Estimation of the Population Correlation Coefficient According to One Set of Samples

(1) **Function**

From the sample correlation coefficient r of one set of sample data of size n, obtain the confidence interval of the population correlation coefficient ρ when the confidence level $1-\alpha$ is specified. The confidence interval (t_1, t_2) is defined as follows.

$$t_1 = \frac{e^{a-b}-1}{e^{a-b}+1}$$

$$t_2 = \frac{e^{a+b}-1}{e^{a+b}+1}$$

where a, b and $\frac{\alpha}{2}$ are as follows.

$$a = \log_e \frac{1+r}{1-r}$$

$$b = \frac{2z_{\frac{\alpha}{2}}}{\sqrt{n-3}}$$

$$\frac{\alpha}{2} = 1 - P(z_{\frac{\alpha}{2}})$$

$P(z)$ is the cumulative distribution function (c.d.f.) of a standard normal distribution.

(2) **Usage**

Double precision:

```
ierr = ASL_d3ietc  (n, r, cl, t);
```

Single precision:

```
ierr = ASL_r3ietc  (n, r, cl, t);
```

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	n	I	1	Input	Number of sample data n
2	r	{ D / R }	1	Input	The sample correlation coefficient r of one set of sample data (See Note (a))
3	cl	{ D / R }	1	Input	Confidence level $100(1-\alpha)(\%)$
4	t	{ D∗ / R∗ }	2	Output	t[0] : Confidence interval lower bound t_1 t[1] : Confidence interval upper bound t_2
5	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) n≥2

(b) $-1.0 < r < 1.0$

(c) 0.0≤cl≤100.0

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
1000	cl=100.0	t[0] = −1.0 and t[1] = 1.0 are performed.
3000	Restriction (a) was not satisfied.	Processing is aborted.
3010	Restriction (b) was not satisfied.	
3020	Restriction (c) was not satisfied.	

(6) **Notes**

(a) The sample correlation coefficient r for n sample data, $\{x_i, y_i\}$ $(i = 1, \cdots, n)$, is defined as follows. (See 4.4.1 $\left\{\begin{matrix}\text{ASL\_d2ccmt}\\ \text{ASL\_r2ccmt}\end{matrix}\right\}$)

$$r = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu_x)(y_i - \mu_y)}{\sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu_x)^2}\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \mu_y)^2}}$$

(7) **Example**

(a) Problem

From the following one set of sample data of size 10, obtain the confidence interval of the population correlation coefficient with 95% confidence level.

x_i	y_i
10.129	63.4
12.611	60.1
13.900	57.2
16.532	46.5
20.822	43.9
26.025	39.6
28.283	39.7
29.199	39.1
30.766	37.8
32.664	27.8

(b) Input data

n=10, cl=95.0 and sample data $\{x_i, y_i\}$.

r : calculate from ASL_d2ccmt.

(c) Main program

```
/*      C interface example for ASL_d3ietc */

#include <stdio.h>
#include <stdlib.h>
#include <asl.h>

int main()
{
    int n;
    double r;
    double cl;
    double t[2];
    int ierr;
    double *a;
    int na;
    int m;
    int nr;
    int ns;
    double *x1;
    double *rr;
    double *wk;
    int isw;
    int kerr;
    int i, j;
    FILE *fp;

    fp = fopen( "d3ietc.dat", "r" );
    if( fp == NULL )
    {
        printf( "file open error\n" );
        return -1;
    }

    printf( "    *** ASL_d3ietc ***\n" );
    printf( "\n     ** Input **\n\n" );

    fscanf( fp, "%lf", &cl );
    fscanf( fp, "%d", &na );
    fscanf( fp, "%d", &n );
    fscanf( fp, "%d", &m );
    fscanf( fp, "%d", &nr );
    fscanf( fp, "%d", &isw );

    a = ( double * )malloc((size_t)( sizeof(double) * (na*m) ));
    if( a == NULL )
    {
        printf( "no enough memory for array a\n" );
        return -1;
    }

    x1 = ( double * )malloc((size_t)( sizeof(double) * m ));
    if( x1 == NULL )
    {
        printf( "no enough memory for array x1\n" );
        return -1;
    }

    rr = ( double * )malloc((size_t)( sizeof(double) * (nr*m) ));
    if( rr == NULL )
    {
        printf( "no enough memory for array rr\n" );
        return -1;
    }

    wk = ( double * )malloc((size_t)( sizeof(double) * m ));
    if( wk == NULL )
    {
        printf( "no enough memory for array wk\n" );
        return -1;
    }

    printf( "\t**  ASL_d2ccmt  **\n");
    printf( "\tisw = %6d\n", isw);
    printf( "\tna  = %6d\n", na);
    printf( "\tn   = %6d\n", n);
    printf( "\tm   = %6d\n", m);
    printf( "\tnr  = %6d\n", nr);

    printf("\n\t sample1 sample2\n");
    for( i=0; i<n; i++ )
    {
        printf("\t");
        for( j=0; j<m; j++ )
        {
            fscanf( fp, "%lf", &a[i+na*j] );
            printf( "%8.3g", a[i+na*j] );
        }
        printf("\n");
```

```
    }
    fclose( fp );

    kerr = ASL_d2ccmt(a, na, n, m, &ns, x1, rr, nr, isw, wk);

    if( kerr != 0 )
    {
        printf("Error occured in ASL_d2ccmt. kerr=%6d\n", kerr);
    return -1;
    }

    r=rr[1];

    printf( "\n\t**  ASL_d3ietc  **\n");
    printf( "\tn   = %6d\n", n);
    printf( "\tr   = %8.3g\n", r);
    printf( "\tcl  = %8.3g\n", cl);

    ierr = ASL_d3ietc(n, r, cl, t);

    printf( "\n     ** Output **\n\n" );
    printf( "\t** ASL_d2ccmt **\n");
    printf( "\tkerr = %6d\n", kerr );
    printf( "\tr    = %8.3g\n", r );
    printf( "\n\t** ASL_d3ietc **\n");
    printf( "\tierr = %6d\n", ierr );

    printf( "\tInterval = (%8.3g, %8.3g)\n", t[0], t[1] );

    free( a );
    free( rr );
    free( x1 );
    free( wk );

    return 0;
}
```

(d) Output results

```
*** ASL_d3ietc ***

 ** Input **

**  ASL_d2ccmt  **
isw =      0
na  =     10
n   =     10
m   =      2
nr  =     10

 sample1 sample2
    10.1    63.4
    12.6    60.1
    13.9    57.2
    16.5    46.5
    20.8    43.9
      26    39.6
    28.3    39.7
    29.2    39.1
    30.8    37.8
    32.7    37.8

**  ASL_d3ietc  **
n   =     10
r   =   -0.945
cl  =       95

 ** Output **

** ASL_d2ccmt **
kerr =      0
r    =   -0.945

** ASL_d3ietc **
ierr =      0
Interval = (  -0.987,   -0.778)
```

6.2.6 ASL_d3iecd, ASL_r3iecd
Interval Estimation of the Difference of the Population Correlation Coefficient According to Two Sets of Independent Samples

(1) **Function**

From the correlation coefficients r_1 and r_2 of two sets of independent sample data of sizes n_1 and n_2, respectively, obtain the confidence interval of the difference of the population correlation coefficients ρ_1 and ρ_2 when the confidence level $1-\alpha$ is specified. If $\rho_1 = \rho_2 = \rho$, the confidence level of ρ is obtained.

(a) When $\rho_1 = \rho_2 = \rho$

The confidence interval (t_1, t_2) of ρ is defined as follows.

$$t_1 = \frac{e^{a-b}-1}{e^{a-b}+1}$$

$$t_2 = \frac{e^{a+b}-1}{e^{a+b}+1}$$

Here,

$$a = \frac{2(n_1-3)z_1 + 2(n_2-3)z_2}{n_1+n_2-6}$$

$$b = \frac{2z_{\frac{\alpha}{2}}}{\sqrt{n_1+n_2-6}}$$

$$z_1 = \frac{1}{2}\log_e \frac{1+r_1}{1-r_1}$$

$$z_2 = \frac{1}{2}\log_e \frac{1+r_2}{1-r_2}$$

$$\frac{\alpha}{2} = 1 - P(z_{\frac{\alpha}{2}})$$

$P(z)$ is the cumulative distribution function (c.d.f.) of a standard normal distribution.

(b) When $\rho_1 \neq \rho_2$

The confidence interval (t_1, t_2) of $\rho_1 - \rho_2$ is defined as follows.

$$t_1 = \frac{e^{a-b}-1}{e^{a-b}+1}$$

$$t_2 = \frac{e^{a+b}-1}{e^{a+b}+1}$$

Here,

$$a = \log_e \frac{1+r_1}{1-r_1} - \log_e \frac{1+r_2}{1-r_2}$$

$$b = 2z_{\frac{\alpha}{2}}\sqrt{\frac{1}{n_1-3} + \frac{1}{n_2-3}}$$

$$\frac{\alpha}{2} = 1 - P(z_{\frac{\alpha}{2}})$$

$P(z)$ is the cumulative distribution function (c.d.f.) of a standard normal distribution.

(2) **Usage**

Double precision:

ierr = ASL_d3iecd (n1, r1, n2, r2, cl, t, isw);

Single precision:

ierr = ASL_r3iecd (n1, r1, n2, r2, cl, t, isw);

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	n1	I	1	Input	Number of first sample data n_1
2	r1	{D / R}	1	Input	The correlation coefficient r_1 of first sample data (See Note (a))
3	n2	I	1	Input	Number of second sample data n_2
4	r2	{D / R}	1	Input	The correlation coefficient r_2 of second sample data (See Note (a))
5	cl	{D / R}	1	Input	Confidence level $100(1-\alpha)(\%)$
6	t	{D∗ / R∗}	3	Output	t[0] : Confidence interval lower bound t_1 t[1] : Confidence interval upper bound t_2
7	isw	I	1	Input	Processing switch isw=1 : when $\rho_1 = \rho_2$ isw=2 : when $\rho_1 \neq \rho_2$
8	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) isw $\in \{1, 2\}$

(b) n1 ≥ 4, n2 ≥ 4

(c) $-1.0 <$ r1 < 1.0, $-1.0 <$ r2 < 1.0

(d) $0.0 \leq$ cl ≤ 100.0

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
1000	cl=100.0	t[0] = −1.0 and t[1] = 1.0 are performed.
3000	Restriction (a) was not satisfied.	Processing is aborted.
3010	Restriction (b) was not satisfied.	
3020	Restriction (c) was not satisfied.	
3030	Restriction (d) was not satisfied.	

(6) **Notes**

(a) The sample correlation coefficient r for n sample data, $\{x_i, y_i\}$ $(i = 1, \cdots, n)$, is defined as follows. (See 4.4.1 $\left\{\begin{matrix}\text{ASL\_d2ccmt}\\ \text{ASL\_r2ccmt}\end{matrix}\right\}$)

$$r = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu_x)(y_i - \mu_y)}{\sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu_x)^2}\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \mu_y)^2}}$$

(7) **Example**

(a) Problem

For the correlation coefficient $r_1 = 0.8$ obtained from a set of 50 observed values and the correlation coefficient $r_2 = 0.6$ obtained from a set of 15 observed values, test the hypothesis $\rho_1 = \rho_2$ related to the population correlation coefficients of the populations to which the respective observed values belong. Assume that the alternative hypothesis is $\rho_1 \neq \rho_2$.

Moreover, if the hypothesis $\rho_1 = \rho_2$ is accepted, obtain the confidence interval of the correlation coefficient of population $\rho_1 = \rho_2 = \rho$ with 95% confidence level. If the hypothesis $\rho_1 = \rho_2$ is rejected, obtain the difference of the two correlation coefficient of population $\rho_1 - \rho_2$ with 95% confidence level.

(b) Input data

n1=50, r1=0.8, n2=40, r2=0.6 and cl=95.0.

(c) Main program

```
/*      C interface example for ASL_d3iecd and STAT_d3tscd */

#include <stdio.h>
#include <asl.h>

int main()
{
    int n1;
    double r1;
    int n2;
    double r2;
    double cl;
    int ir;
    double t[2];
    double z[2];
    int isw_ie;
    int isw_ts;
    int ierr_ie;
    int ierr_ts;
    int i;


    printf( "    *** ASL_d3tscd, STAT_d3iecd ***\n" );
    printf( "\n     ** Input **\n\n" );
```

```
    isw_ts=1;
    n1=50; r1=0.8e0;
    n2=40; r2=0.6e0;
    cl=95.0e0;

    printf( "\t** ASL_d3tscd **\n");
    printf( "\tisw_ts = %6d\n", isw_ts );
    printf( "\tn1     = %6d\n", n1 );
    printf( "\tn2     = %6d\n", n2 );
    printf( "\tr1     = %8.3g\n", r1 );
    printf( "\tr2     = %8.3g\n", r2 );
    printf( "\tcl     = %8.3g\n", cl );

ierr_ts = ASL_d3tscd(n1, r1, n2, r2, cl, &ir, z, isw_ts);

    if ( ir == 0 ){
        isw_ie = 1;
    }else{
        isw_ie = 2;
    }

    printf( "\n\t** ASL_d3iecd **\n");
    printf( "\tisw_ie = %6d\n", isw_ie );
    printf( "\tn1     = %6d\n", n1 );
    printf( "\tn2     = %6d\n", n2 );
    printf( "\tr1     = %8.3g\n", r1 );
    printf( "\tr2     = %8.3g\n", r2 );
    printf( "\tcl     = %8.3g\n", cl );

ierr_ie = ASL_d3iecd(n1, r1, n2, r2, cl, t, isw_ie);

printf( "\n     ** Output **\n\n" );

printf( "\t** ASL_d3tscd **\n" );
printf( "\tierr_ts = %6d\n\n", ierr_ts );

if ( ir == 0 ){
    printf( "\tHypothesis, rho1 = rho2, is accepted.\n " );
}else{
    printf( "\tHypothesis, rho1 = rho2, is rejected.\n " );
}
printf( "\n" );
for( i=0 ; i<2 ; i++ )
{
    printf( "\tz[%2d] = %8.3g\n", i, z[i] );
}

printf( "\n\t** ASL_d3iecd **\n" );
printf( "\tierr_ie = %6d\n\n", ierr_ie );

if( isw_ie == 1 ){
    printf( "\tInterval of rho = (%8.3g, %8.3g)\n", t[0], t[1] );
}else{
    printf( "\tInterval of rho1 - rho2 = (%8.3g, %8.3g)\n", t[0], t[1] );
}

    return 0;
}
```

(d) Output results

```
*** ASL_d3tscd, STAT_d3iecd ***

 ** Input **

** ASL_d3tscd **
isw_ts =      1
n1     =     50
n2     =     40
r1     =      0.8
r2     =      0.6
cl     =       95

** ASL_d3iecd **
isw_ie =      1
n1     =     50
n2     =     40
r1     =      0.8
r2     =      0.6
cl     =       95

 ** Output **

** ASL_d3tscd **
ierr_ts =      0

Hypothesis, rho1 = rho2, is accepted.

z[ 0] =     1.84
z[ 1] =     1.96
```

```
** ASL_d3iecd **
ierr_ie =      0

Interval of rho = (   0.608,    0.812)
```

6.2.7 ASL_d3iesr, ASL_r3iesr
Interval Estimation in the Simple Linear Regression

(1) **Function**

For regression coefficient a, constant term b, and a given specific data x_0 in the following simple linear regression expression (or regression line) related to one set of sample data $\{x_i, y_i\}$ $(1, \cdots, n)$ of size n

$$\hat{y}_i = ax_i + b$$

obtain the estimate $\hat{y}_0$ and the confidence interval of the confidence level $1 - \alpha$ of the theoretical value $Ax_0 - B$. Assume that y_i corresponding to each x_i is the random sample from the normal population having the mean $Ax_i - B$ and the variance σ^2. Obtain the regression coefficient a and constant term b of the sample data from the following normal equations.

$$\begin{cases} \sum_{i=1}^{n} y_i = a\sum_{i=1}^{n} x_i + bn \\ \sum_{i=1}^{n} x_i y_i = a\sum_{i=1}^{n} x_i^2 + b\sum_{i=1}^{n} x_i \end{cases}$$

The confidence interval (t_1, t_2) is defined as follows.

(a) Regression coefficient

$$t_1 = a - t_{\frac{\alpha}{2}} \cdot s_a$$

$$t_2 = a + t_{\frac{\alpha}{2}} \cdot s_a$$

Here,

$$s_a = \sqrt{\frac{\sigma^2}{\sum (x_i - \mu_x)^2}}$$

i. When the population variance is known

σ^2 : Population variance

$$\frac{\alpha}{2} = 1 - P(t_{\frac{\alpha}{2}})$$

Here, $P(t)$ is the cumulative distribution function (c.d.f.) of a standard normal distribution.

ii. When the population variance is unknown

σ^2 : Unbiased variance of error variation

$$\frac{\alpha}{2} = 1 - P(t_{\frac{\alpha}{2}}|n-2)$$

Here, $P(t|n)$ is the cumulative distribution function (c.d.f.) of a t distribution having number of degrees of freedom n.

(b) Constant term

$$t_1 = a - t_{\frac{\alpha}{2}} \cdot s_b$$

$$t_2 = a + t_{\frac{\alpha}{2}} \cdot s_b$$

Here,

$$s_b = \sqrt{\sigma^2 \left[\frac{1}{n} + \frac{\mu_x^2}{\sum (x_i - \mu_x)^2}\right]}$$

i. When the population variance is known

σ^2 : Population variance

$$\frac{\alpha}{2} = 1 - P(t_{\frac{\alpha}{2}})$$

Here, $P(t)$ is the cumulative distribution function (c.d.f.) of a standard normal distribution.

ii. When the population variance is unknown

σ^2 : Unbiased variance of error variation

$$\frac{\alpha}{2} = 1 - P(t_{\frac{\alpha}{2}}|n-2)$$

Here, $P(t|n)$ is the cumulative distribution function (c.d.f.) of a t distribution having number of degrees of freedom n.

(c) Estimated value

$$t_1 = \hat{y}_0 - t_{\frac{\alpha}{2}} \cdot s_y$$

$$t_2 = \hat{y}_0 + t_{\frac{\alpha}{2}} \cdot s_y$$

Here,

$$s_y = \sqrt{\sigma^2 \left[1 + \frac{1}{n} + \frac{(x_0 - \mu_x)^2}{\sum (x_i - \mu_x)^2}\right]}$$

i. When the population variance is known

σ^2 : Population variance

$$\frac{\alpha}{2} = 1 - P(t_{\frac{\alpha}{2}})$$

Here, $P(t)$ is the cumulative distribution function (c.d.f.) of a standard normal distribution.

ii. When the population variance is unknown

σ^2 : Unbiased variance of error variation

$$\frac{\alpha}{2} = 1 - P(t_{\frac{\alpha}{2}}|n-2)$$

Here, $P(t|n)$ is the cumulative distribution function (c.d.f.) of a t distribution having number of degrees of freedom n.

(d) Theoretical value

$$t_1 = \hat{y}_0 - t_{\frac{\alpha}{2}} \cdot s_0$$

$$t_2 = \hat{y}_0 + t_{\frac{\alpha}{2}} \cdot s_0$$

Here,

$$s_0 = \sqrt{\sigma^2 \left[\frac{1}{n} + \frac{(x_0 - \mu_x)^2}{\sum (x_i - \mu_x)^2}\right]}$$

i. When the population variance is known

σ^2 : Population variance

$$\frac{\alpha}{2} = 1 - P(t_{\frac{\alpha}{2}})$$

Here, $P(t)$ is the cumulative distribution function (c.d.f.) of a standard normal distribution.

ii. When the population variance is unknown

σ^2 : Unbiased variance of error variation

$$\frac{\alpha}{2} = 1 - P(t_{\frac{\alpha}{2}}|n-2)$$

Here, $P(t|n)$ is the cumulative distribution function (c.d.f.) of a t distribution having number of degrees of freedom n.

(2) **Usage**

Double precision:

ierr = ASL_d3iesr (x, n, y, &yv, x0, cl, t, stat, isw1, isw2, w);

Single precision:

ierr = ASL_r3iesr (x, n, y, &yv, x0, cl, t, stat, isw1, isw2, w);

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	x	{D∗ / R∗}	n	Input	Values of independent variable X of sample, $x_i(i = 1, n)$
2	n	I	1	Input	Number of sample data n
3	y	{D∗ / R∗}	n	Input	Values of dependent variable Y of sample, $y_i(i = 1, n)$
4	yv	{D∗ / R∗}	1	Input	Variance of the population to which dependent variable Y belongs(if isw2 = 1)
				Output	Unbiased variance of error variation σ^2 (if isw2 = 2) (See 10.2.1)
5	x0	{D / R}	1	Input	Value of x_0 to obtain the estimated value or the theoretical value (If isw1 = 1 or isw1 = 2, initialization is not necessary.)
6	cl	{D / R}	1	Input	Confidence level $100(1 - \alpha)(\%)$
7	t	{D∗ / R∗}	2	Output	t[0] : Confidence interval lower bound t_1 t[1] : Confidence interval upper bound t_2
8	stat	{D∗ / R∗}	2	Output	stat[0] : The sample regression coefficient stat[1] : The sample constant term

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
9	isw1	I	1	Input	The switch for selection of the statistic. isw1=1 : To obtain the confidence interval of the regression coefficient. isw1=2 : To obtain the confidence interval of the constant term. isw1=3 : To obtain the confidence interval of the estimated value. isw1=4 : To obtain the confidence interval of the theoretical value.
10	isw2	I	1	Input	Switch for variance isw2=1 : The variance of the population is entered for yv isw2=2 : The variance (not an unbiased estimate) of the sample data is entered for yv
11	w	$\left\{\begin{matrix}\text{D}* \\ \text{R}*\end{matrix}\right\}$	29	Work	Work area
12	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) isw1 $\in \{1, 2, 3, 4\}$

(b) isw2 $\in \{1, 2\}$

(c) n ≥ 3

(d) $0.0 \leq$ cl ≤ 100.0

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
1000	cl=100.0	The negative minimum value is set for t[0] and the positive maximum value is set for t[1].
3000	Restriction (a) was not satisfied.	Processing is aborted.
3010	Restriction (b) was not satisfied.	
3020	Restriction (c) was not satisfied.	
3030	Restriction (d) was not satisfied.	
4000	There are no difference among independent variable X.	
4100	The Unbiased variance of error variation is 0.0. (See 10.2.1)	

(6) **Notes**

None

(7) **Example**

(a) Problem

From the following one set of sample data of size 9, obtain the confidence interval regression coefficient of population with 95% confidence level.

x_i	y_i
1	3
2	3
3	5
4	5
5	6
6	7
7	8
8	8
9	9

The variance value of the population to which the samples belongs is unknown.

(b) Input data

isw1=1, isw2=2 n=9, array x, array y and cl=95.0.

(c) Main program

```
/*      C interface example for ASL_d3iesr */

#include <stdio.h>
#include <stdlib.h>
#include <asl.h>

int main()
{
    int n;
    double cl;
    double yv;
    double x0;
    double t[2];
    double stat[2];
    double w[29];
    int isw1;
    int isw2;
    int ierr;
    double *x;
    double *y;
    int i;
    FILE *fp;

    fp = fopen( "d3iesr.dat", "r" );
    if( fp == NULL )
    {
        printf( "file open error\n" );
        return -1;
    }

    printf( "    *** ASL_d3iesr ***\n" );
    printf( "\n     ** Input **\n\n" );

    fscanf( fp, "%d", &isw1);
    fscanf( fp, "%d", &isw2);
    fscanf( fp, "%d", &n );
    fscanf( fp, "%lf", &cl );
    fscanf( fp, "%lf", &x0 );

    x = ( double * )malloc((size_t)( sizeof(double) * (n) ));
    if( x == NULL )
    {
        printf( "no enough memory for array x\n" );
        return -1;
    }

    y = ( double * )malloc((size_t)( sizeof(double) * (n) ));
    if( y == NULL )
    {
        printf( "no enough memory for array y\n" );
        return -1;
    }
```

```
    for( i=0; i<n; i++ )
    {
            fscanf( fp, "%lf", &x[i] );
    }
    for( i=0; i<n; i++ )
    {
            fscanf( fp, "%lf", &y[i] );
    }
    fclose( fp );

    printf( "\n\t**  ASL_d3iesr  **\n");
    printf( "\tisw1    = %6d\n", isw1);
    printf( "\tisw2    = %6d\n", isw2);
    printf( "\tn       = %6d\n", n);
    printf( "\tx0      = %8.3g\n", x0);
    printf( "\tcl      = %8.3g\n", cl);
    printf("\n\t sample1  sample2\n");
    for( i=0; i<n; i++ )
    {
        printf( "\t%8.3g %8.3g\n", x[i], y[i]);
    }

    ierr = ASL_d3iesr(x, n, y, &yv, x0, cl, t, stat, isw1, isw2, w);

    printf( "\n     ** Output **\n\n" );
    printf( "\tierr = %6d\n\n", ierr );

    if( isw1 == 1 ){
        printf( "\tInterval of regression coefficient = (%8.3g, %8.3g)\n", t[0], t[1] );
    }else if( isw1 == 2 ){
        printf( "\tInterval of constant term = (%8.3g, %8.3g)\n", t[0], t[1] );
    }else if( isw1 == 3 ){
        printf( "\tInterval of predictive value = (%8.3g, %8.3g)\n", t[0], t[1] );
    }else if( isw1 == 4 ){
        printf( "\tInterval theoretical value = (%8.3g, %8.3g)\n", t[0], t[1] );
    }
    printf( "\n" );
    printf( "\tRegression coefficient of sample = %8.3g\n", stat[0] );
    printf( "\tConstant term of sample          = %8.3g\n", stat[1] );

    free( x );
    free( y );

    return 0;
}
```

(d) Output results

```
*** ASL_d3iesr ***

 ** Input **


**  ASL_d3iesr  **
isw1    =       1
isw2    =       2
n       =       9
x0      =         5
cl      =        95

 sample1  sample2
       1        3
       2        3
       3        5
       4        5
       5        6
       6        7
       7        8
       8        8
       9        9

 ** Output **

ierr =       0

Interval of regression coefficient = (   0.658,    0.909)

Regression coefficient of sample =    0.783
Constant term of sample          =     2.08
```

6.3 TESTS

6.3.1 ASL_d3tsra, ASL_r3tsra
Test of the Population Ratio According to One Set of Samples

(1) **Function**

When the number of data having the observed characteristic in the one set of sample data of size n is m, test the hypothesis $p = p_0$ related to the ratio p in the population to which the respective set of sample data belong with the confidence level $1 - \alpha$. The test criteria are as follows.

(a) When the alternative hypothesis is $p \neq p_0$

For f_1 and f_2 defined as follows

$$f_1 = \frac{2(n-m)p_0}{2(m+1)(1-p_0)}$$

$$f_2 = \frac{2m(1-p_0)}{2(n-m+1)p_0}$$

$$\begin{cases} \text{If } f_1 \geq F_1 \text{ or } f_2 \geq F_2, \text{ reject} \\ \text{If } f_1 < F_1 \text{ and } f_2 < F_2, \text{ accept} \end{cases}$$

Here,

$$\frac{\alpha}{2} = 1 - P(F_1|2(n-m+1), 2m) = 1 - P(F_2|2(m+1), 2(n-m))$$

$P(F|n_1, n_2)$ is the cumulative distribution function (c.d.f.) of a F distribution having numbers of degrees of freedom n_1 and n_2.

(b) When the alternative hypothesis is $p < p_0$

For f_1 defined as follows

$$f_1 = \frac{2(n-m)p_0}{2(m+1)(1-p_0)}$$

$$\begin{cases} \text{If } f_1 \geq F_1^*, \text{ reject} \\ \text{If } f_1 < F_1^*, \text{ accept} \end{cases} \quad \text{Here,}$$

$$\alpha = 1 - P(F_1^*|2(m+1), 2(n-m))$$

$P(F|n_1, n_2)$ is the cumulative distribution function (c.d.f.) of a F distribution having numbers of degrees of freedom n_1 and n_2.

(c) When the alternative hypothesis is $p > p_0$

For f_2 defined as follows

$$f_2 = \frac{2m(1-p_0)}{2(n-m+1)(1-p_0)}$$

$$\begin{cases} \text{If } f_2 \geq F_2^*, \text{ reject} \\ \text{If } f_2 < F_2^*, \text{ accept} \end{cases} \quad \text{Here,}$$

$$\alpha = 1 - P(F_2^*|2(n-m+1), 2m)$$

$P(F|n_1, n_2)$ is the cumulative distribution function (c.d.f.) of a F distribution having numbers of degrees of freedom n_1 and n_2.

(2) **Usage**

Double precision:

ierr = ASL_d3tsra (n, m, cl, p0, &ir, f, isw);

Single precision:

ierr = ASL_r3tsra (n, m, cl, p0, &ir, f, isw);

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	n	I	1	Input	Number of sample data n
2	m	I	1	Input	Number of samples having the characteristic being observed m
3	cl	{D / R}	1	Input	Confidence level $100(1-\alpha)(\%)$
4	p0	{D / R}	1	Input	Tested ratio value p_0
5	ir	I*	1	Output	Test result ir=0 : Hypothesis $p = p_0$ is accepted ir=1 : Hypothesis $p = p_0$ is rejected
6	f	{D* / R*}	4	Output	When isw=1 f[0] : Value of f_1 f[1] : Value of f_2 f[2] : Value of F distribution F_1 f[3] : Value of F distribution F_2 When isw=2 f[0] : Value of f_1 f[1] : Value of F distribution F_1^* When isw=3 f[0] : Value of f_2 f[1] : Value of F distribution F_2^*
7	isw	I	1	Input	The alternative hypothesis switch isw=1 : When the alternative hypothesis is $p \neq p_0$ isw=2 : When the alternative hypothesis is $p > p_0$ isw=3 : When the alternative hypothesis is $p < p_0$
8	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) isw $\in \{1, 2, 3\}$

(b) n > 0

(c) $0 <$ m $<$ n

(d) $0.0 \leq$ p0 ≤ 1.0

(e) $0.0 \leq$ cl ≤ 100.0

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
1000	isw=1 : cl=100.0 isw=2 or 3 : cl=0.0 or cl=100.0.	isw=1 : 0.0 or the positive maximum value is set for f[2] and f[3]. isw=2 or 3 : 0.0 or the positive maximum value is set for f[1].
3000	Restriction (a) was not satisfied.	Processing is aborted.
3010	Restriction (b) was not satisfied.	
3020	Restriction (c) was not satisfied.	
3030	Restriction (d) was not satisfied.	
3040	Restriction (e) was not satisfied.	

(6) **Notes**

None

(7) **Example**

(a) Problem

When the number of sample data having the observed characteristic in the independent sample of size 19 is 5, test the hypothesis $p = 0.5$ related to the population ratio p with 95% confidence level. Assume that the alternative hypothesis is $p \neq 0.5$.

(b) Input data

isw=1, n=19, m=5, p=0.5 and cl=95.0.

(c) Main program

```
/*      C interface example for ASL_d3tsra */

#include <stdio.h>
#include <stdlib.h>
#include <asl.h>

int main()
{
    int n;
    int m;
    double p0;
    double cl;
    int ir;
    double f[4];
    int isw;
    int ierr;
    int i;
    int nf;


    printf( "    *** ASL_d3tsra ***\n" );
```

```
    printf( "\n     ** Input **\n\n" );

    n=19;
    m=5;
    p0=0.5e0;
    cl=95.0e0;
    isw=1;
    nf=4;

    printf( "\tisw = %6d\n", isw);
    printf( "\tn   = %6d\n", n);
    printf( "\tm   = %6d\n", m);
    printf( "\tp0  = %8.3g\n", p0);
    printf( "\tcl  = %8.3g\n", cl);

    ierr = ASL_d3tsra(n, m, cl, p0, &ir, f, isw);

    printf( "\n     ** Output **\n\n" );
    printf( "\tierr  = %4d\n\n", ierr );

    if ( ir == 0 ){
        printf( "\tHypothesis, p = %8.3g, is accepted.\n ", p0 );
    }else{
        printf( "\tHypothesis, p = %8.3g, is rejected.\n ", p0 );
    }

    printf("\n");
    for( i=0 ; i<nf/2 ; i++ )
    {
        printf( "\tf[%2d] = %8.3g\tf[%2d] = %8.3g\n", i, f[i], i+2, f[i+2] );
    }

    return 0;
}
```

(d) Output results

```
*** ASL_d3tsra ***

 ** Input **

isw =      1
n   =     19
m   =      5
p0  =      0.5
cl  =       95

 ** Output **

ierr  =    0

Hypothesis, p =      0.5, is accepted.

f[ 0] =     2.33    f[ 2] =     2.45
f[ 1] =    0.333    f[ 3] =     3.31
```

6.3.2 ASL_d3tsrd, ASL_r3tsrd
Test of the Difference of the Population Ratios According to Two Sets of Independent Samples

(1) **Function**

When the number of data having the observed characteristic in the two sets of independent sample data of sizes n_1 and n_2 are m_1 and m_2, respectively, test the hypothesis $p_1 = p_2$ related to the ratios p_1 and p_2 in the population to which the respective sets of sample data belong with the confidence level $1 - \alpha$.
Let the sample ratios in the two sets of independent samples, respectively, be $\hat{p}_1$ and $\hat{p}_2$ as shown below.

$$\hat{p}_1 = \frac{m_1}{n_1}, \hat{p}_2 = \frac{m_2}{n_2}$$

The test criteria are as follows.

(a) When no continuity correction is performed

i. When the alternative hypothesis is $p_1 \neq p_2$

For z defined as follows

$$z = \frac{\hat{p}_1 - \hat{p}_2}{\sqrt{\hat{p}(1-\hat{p})(\frac{1}{n_1} + \frac{1}{n_2})}}$$

$$\begin{cases} \text{If } |z| \geq z_{\frac{\alpha}{2}}, \text{ reject} \\ \text{If } |z| < z_{\frac{\alpha}{2}}, \text{ accept} \end{cases}$$

where, $\hat{p}$ and $\frac{\alpha}{2}$ are as follows.

$$\hat{p} = \frac{n_1\hat{p}_1 + n_2\hat{p}_2}{n_1 + n_2} = \frac{m_1 + m_2}{n_1 + n_2}$$

$$\frac{\alpha}{2} = 1 - P(z_{\frac{\alpha}{2}})$$

Here, $P(z)$ is the c.d.f. value of the standard normal distribution.

ii. When the alternative hypothesis is $p_1 < p_2$

For z defined as follows

$$z = \frac{\hat{p}_1 - \hat{p}_2}{\sqrt{\hat{p}(1-\hat{p})(\frac{1}{n_1} + \frac{1}{n_2})}}$$

$$\begin{cases} \text{If } z \leq -z_{\alpha}, \text{ reject} \\ \text{If } z > -z_{\alpha}, \text{ accept} \end{cases}$$

where, $\hat{p}$ and α are as follows.

$$\hat{p} = \frac{n_1\hat{p}_1 + n_2\hat{p}_2}{n_1 + n_2} = \frac{m_1 + m_2}{n_1 + n_2}$$

$$\alpha = 1 - P(z_{\alpha})$$

Here, $P(z)$ is the c.d.f. value of the standard normal distribution.

iii. When the alternative hypothesis is $p_1 > p_2$

For z defined as follows

$$z = \frac{\hat{p}_1 - \hat{p}_2}{\sqrt{\hat{p}(1-\hat{p})(\frac{1}{n_1} + \frac{1}{n_2})}}$$

$$\begin{cases} \text{If } z \geq z_{\alpha}, \text{ reject} \\ \text{If } z < z_{\alpha}, \text{ accept} \end{cases}$$

where, $\hat{p}$ and α are as follows.

$$\hat{p} = \frac{n_1\hat{p}_1 + n_2\hat{p}_2}{n_1 + n_2} = \frac{m_1 + m_2}{n_1 + n_2}$$

$$\alpha = 1 - P(z_\alpha)$$

Here, $P(z)$ is the c.d.f. value of the standard normal distribution.

(b) When continuity correction is performed

i. When the alternative hypothesis is $p_1 \neq p_2$

For z defined as follows

$$\begin{cases} z = \dfrac{\hat{p}_1 - \hat{p}_2 - 0.5(\frac{1}{n_1} + \frac{1}{n_2})}{\sqrt{\hat{p}(1-\hat{p})(\frac{1}{n_1} + \frac{1}{n_2})}} (\text{when } \hat{p}_1 \geq \hat{p}_2) \\ z = \dfrac{\hat{p}_1 - \hat{p}_2 + 0.5(\frac{1}{n_1} + \frac{1}{n_2})}{\sqrt{\hat{p}(1-\hat{p})(\frac{1}{n_1} + \frac{1}{n_2})}} (\text{when } \hat{p}_1 < \hat{p}_2) \end{cases}$$

$$\begin{cases} \text{If } |z| \geq z_{\frac{\alpha}{2}}, \text{ reject} \\ \text{If } |z| < z_{\frac{\alpha}{2}}, \text{ accept} \end{cases}$$

where, $\hat{p}$ and $\frac{\alpha}{2}$ are as follows.

$$\hat{p} = \frac{n_1\hat{p}_1 + n_2\hat{p}_2}{n_1 + n_2} = \frac{m_1 + m_2}{n_1 + n_2}$$

$$\frac{\alpha}{2} = 1 - P(z_{\frac{\alpha}{2}})$$

Here, $P(z)$ is the c.d.f. value of the standard normal distribution.

ii. When the alternative hypothesis is $p_1 < p_2$

For z defined as follows

$$\begin{cases} z = \dfrac{\hat{p}_1 - \hat{p}_2 - 0.5(\frac{1}{n_1} + \frac{1}{n_2})}{\sqrt{\hat{p}(1-\hat{p})(\frac{1}{n_1} + \frac{1}{n_2})}} (\text{when } \hat{p}_1 \geq \hat{p}_2) \\ z = \dfrac{\hat{p}_1 - \hat{p}_2 + 0.5(\frac{1}{n_1} + \frac{1}{n_2})}{\sqrt{\hat{p}(1-\hat{p})(\frac{1}{n_1} + \frac{1}{n_2})}} (\text{when } \hat{p}_1 < \hat{p}_2) \end{cases}$$

$$\begin{cases} \text{If } z \leq -z_\alpha, \text{ reject} \\ \text{If } z > -z_\alpha, \text{ accept} \end{cases}$$

where, $\hat{p}$ and α are as follows.

$$\hat{p} = \frac{n_1\hat{p}_1 + n_2\hat{p}_2}{n_1 + n_2} = \frac{m_1 + m_2}{n_1 + n_2}$$

$$\alpha = 1 - P(z_\alpha)$$

Here, $P(z)$ is the c.d.f. value of the standard normal distribution.

iii. When the alternative hypothesis is $p_1 > p_2$

For z defined as follows

$$\begin{cases} z = \dfrac{\hat{p}_1 - \hat{p}_2 - 0.5(\frac{1}{n_1} + \frac{1}{n_2})}{\sqrt{\hat{p}(1-\hat{p})(\frac{1}{n_1} + \frac{1}{n_2})}} (\text{when } \hat{p}_1 \geq \hat{p}_2) \\ z = \dfrac{\hat{p}_1 - \hat{p}_2 + 0.5(\frac{1}{n_1} + \frac{1}{n_2})}{\sqrt{\hat{p}(1-\hat{p})(\frac{1}{n_1} + \frac{1}{n_2})}} (\text{when } \hat{p}_1 < \hat{p}_2) \end{cases}$$

$$\begin{cases} \text{If } z \geq z_\alpha, \text{ reject} \\ \text{If } z < z_\alpha, \text{ accept} \end{cases}$$

where, $\hat{p}$ and α are as follows.

$$\hat{p} = \frac{n_1\hat{p}_1 + n_2\hat{p}_2}{n_1 + n_2} = \frac{m_1 + m_2}{n_1 + n_2}$$

$$\alpha = 1 - P(z_\alpha)$$

Here, $P(z)$ is the c.d.f. value of the standard normal distribution.

(2) **Usage**

Double precision:

```
ierr = ASL_d3tsrd  (n1, m1, n2, m2, cl, &ir, z, isw1, isw2);
```

Single precision:

```
ierr = ASL_r3tsrd  (n1, m1, n2, m2, cl, &ir, z, isw1, isw2);
```

(3) **Arguments and Return Value**

D:Double precision real　Z:Double precision complex　I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real　C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	n1	I	1	Input	Number of first sample data n_1
2	m1	I	1	Input	Number of data having the observed characteristic in first sample data m_1
3	n2	I	1	Input	Number of second sample data n_2
4	m2	I	1	Input	Number of data having the observed characteristic in second sample data m_2
5	cl	{D / R}	1	Input	Confidence level $100(1-\alpha)(\%)$
6	ir	I*	1	Output	Test result ir=0 : Hypothesis $p_1 = p_2$ is accepted ir=1 : Hypothesis $p_1 = p_2$ is rejected
7	z	{D* / R*}	2	Output	When isw2=1 z[0] : Values of z z[1] : Values of a standard normal distribution $z_{\frac{\alpha}{2}}$ When isw2=2 z[0] : Values of z z[1] : Values of a standard normal distribution $-z_\alpha$ When isw2=3 z[0] : Values of z z[1] : Values of a standard normal distribution z_α
8	isw1	I	1	Input	Processing switch isw1=1 : Continuity correction is not performed isw1=2 : Continuity correction is performed

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
9	isw2	I	1	Input	The alternative hypothesis switch isw2=1 : When the alternative hypothesis is $p \neq p_0$ isw2=2 : When the alternative hypothesis is $p > p_0$ isw2=3 : When the alternative hypothesis is $p < p_0$
10	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) isw1 $\in \{1, 2\}$

(b) isw2 $\in \{1, 2, 3\}$

(c) n1 > 0, n2 > 0

(d) $0 <$ m1 $<$ n1, $0 <$ m2 $<$ n2

(e) $0.0 \leq$ cl ≤ 100.0

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
1000	isw2=1 : cl=100.0 isw2=2 or 3 : cl=0.0 or cl=100.0.	isw2=1 : The positive maximum value is set for z[1]. isw2=2 or 3 : The positive maximum value or the negative minimum value is set for z[1].
3000	Restriction (a) was not satisfied.	Processing is aborted.
3010	Restriction (b) was not satisfied.	
3020	Restriction (c) was not satisfied.	
3030	Restriction (d) was not satisfied.	
3040	Restriction (e) was not satisfied.	

(6) **Notes**

None

(7) **Example**

(a) Problem

When the numbers of data having the observed characteristic in the two sets of independent samples of size 200 and 250 are 140 and 150, respectively, test the hypothesis $p_1 = p_2$ related to the population ratios p_1 and p_2 with 95% confidence level and perform continuity correction. Assume that the alternative hypothesis is $p_1 \neq p_2$.

(b) Input data

isw1=2, isw2=1 n1=200, m1=140, n2=250, m2=150 and cl=95.0.

(c) Main program

```
/*      C interface example for ASL_d3tsrd */

#include <stdio.h>
#include <stdlib.h>
#include <asl.h>

int main()
{
    int n1;
    int m1;
    int n2;
    int m2;
    double cl;
    int ir;
    double z[2];
    int isw1;
    int isw2;
    int ierr;
    int i;


    printf( "    *** ASL_d3tsrd ***\n" );
    printf( "\n     ** Input **\n\n" );

    isw1=2;
    isw2=1;
    n1=200;
    m1=140;
    n2=250;
    m2=150;
    cl=95.0e0;

    printf( "\tisw1= %6d\n", isw1);
    printf( "\tisw2= %6d\n", isw2);
    printf( "\tn1  = %6d\n", n1);
    printf( "\tm1  = %6d\n", m1);
    printf( "\tn2  = %6d\n", n2);
    printf( "\tm2  = %6d\n", m2);
    printf( "\tcl  = %8.3g\n", cl);

    ierr = ASL_d3tsrd(n1, m1, n2, m2, cl, &ir, z, isw1, isw2);

    printf( "\n     ** Output **\n\n" );
    printf( "\tierr  = %4d\n\n", ierr );

    if ( ir == 0 ){
        printf( "\tHypothesis, p1 = p2, is accepted.\n " );
    }else{
        printf( "\tHypothesis, p1 = p2, is rejected.\n " );
    }

    printf("\n");
    for( i=0 ; i<2 ; i++ )
    {
        printf( "\tz[%2d] = %8.3g\n", i, z[i] );
    }

    return 0;
}
```

(d) Output results

```
*** ASL_d3tsrd ***

 ** Input **

isw1=      2
isw2=      1
n1  =    200
m1  =    140
n2  =    250
m2  =    150
cl  =       95

 ** Output **

ierr  =    0

Hypothesis, p1 = p2, is rejected.

z[ 0] =      2.1
z[ 1] =     1.96
```

6.3.3 ASL_d3tsme, ASL_r3tsme
Test of the Population Mean According to One Set of Samples

(1) **Function**

From the mean μ_x and variance (or population variance) σ^2 of one set of sample data of size n, test the hypothesis $\mu = \mu_0$ with the confidence level $1 - \alpha$. The test criteria are as follows.

(a) When the population variance is known

i. When the alternative hypothesis is $\mu \neq \mu_0$

For z defined as follows

$$z = \frac{\mu_x - \mu_0}{\sqrt{\frac{\sigma^2}{n}}}$$

$$\begin{cases} \text{If } |z| \geq z_{\frac{\alpha}{2}}, \text{ reject} \\ \text{If } |z| < z_{\frac{\alpha}{2}}, \text{ accept} \end{cases}$$

where,

σ^2 : Population variance

$$\frac{\alpha}{2} = 1 - P(z_{\frac{\alpha}{2}})$$

$P(z)$ is the cumulative distribution function (c.d.f.) of a standard normal distribution.

ii. When the alternative hypothesis is $\mu < \mu_0$

For z defined as follows

$$z = \frac{\mu_x - \mu_0}{\sqrt{\frac{\sigma^2}{n}}}$$

$$\begin{cases} \text{If } z \leq -z_{\alpha}, \text{ reject} \\ \text{If } z > -z_{\alpha}, \text{ accept} \end{cases}$$

where,

σ^2 : Population variance

$$\alpha = 1 - P(z_{\alpha})$$

$P(z)$ is the cumulative distribution function (c.d.f.) of a standard normal distribution.

iii. When the alternative hypothesis is $\mu > \mu_0$

For z defined as follows

$$z = \frac{\mu_x - \mu_0}{\sqrt{\frac{\sigma^2}{n}}}$$

$$\begin{cases} \text{If } z \geq z_{\alpha}, \text{ reject} \\ \text{If } z < z_{\alpha}, \text{ accept} \end{cases}$$

where,

σ^2 : Population variance

$$\alpha = 1 - P(z_{\alpha})$$

$P(z)$ is the cumulative distribution function (c.d.f.) of a standard normal distribution.

(b) When the population variance is unknown

i. When the alternative hypothesis is $\mu \neq \mu_0$

For t defined as follows

$$t = \frac{\mu_x - \mu_0}{\sqrt{\frac{\sigma^2}{n}}}$$

$$\begin{cases} \text{If } |t| \geq t_{\frac{\alpha}{2}}, \text{ reject} \\ \text{If } |t| < t_{\frac{\alpha}{2}}, \text{ accept} \end{cases}$$

where,

σ^2 : Unbiased estimate of population variance

$$\frac{\alpha}{2} = 1 - P(t_{\frac{\alpha}{2}}|n-1)$$

Here, $P(t|n)$ is the cumulative distribution function (c.d.f.) of a t distribution having number of degrees of freedom n.

ii. When the alternative hypothesis is $\mu < \mu_0$

For t defined as follows

$$t = \frac{\mu_x - \mu_0}{\sqrt{\frac{\sigma^2}{n}}}$$

$$\begin{cases} \text{If } t \leq -t_\alpha, \text{ reject} \\ \text{If } t > -t_\alpha, \text{ accept} \end{cases}$$

where,

σ^2 : Unbiased estimate of population variance

$$\alpha = 1 - P(t_\alpha|n-1)$$

Here, $P(t|n)$ is the cumulative distribution function (c.d.f.) of a t distribution having number of degrees of freedom n.

iii. When the alternative hypothesis is $\mu > \mu_0$ For t defined as follows

$$t = \frac{\mu_x - \mu_0}{\sqrt{\frac{\sigma^2}{n}}}$$

$$\begin{cases} \text{If } t \geq t_\alpha, \text{ reject} \\ \text{If } t < t_\alpha, \text{ accept} \end{cases}$$

where,

σ^2 : Unbiased estimate of population variance

$$\alpha = 1 - P(t_\alpha|n-1)$$

Here, $P(t|n)$ is the cumulative distribution function (c.d.f.) of a t distribution having number of degrees of freedom n.

(2) **Usage**

Double precision:

```
ierr = ASL_d3tsme  (n, xe, xv, cl, xi, &ir, z, isw1, isw2);
```

Single precision:

```
ierr = ASL_r3tsme  (n, xe, xv, cl, xi, &ir, z, isw1, isw2);
```

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	n	I	1	Input	Number of sample data n
2	xe	{D / R}	1	Input	Mean of sample data $\bar{x}$
3	xv	{D / R}	1	Input	Variance of sample data or population σ^2
4	cl	{D / R}	1	Input	Confidence level $100(1-\alpha)(\%)$
5	xi	{D / R}	1	Input	Tested population mean μ_0
6	ir	I*	1	Output	Test result ir=0 : Hypothesis $\mu = \mu_0$ is accepted ir=1 : Hypothesis $\mu = \mu_0$ is rejected
7	z	{D* / R*}	2	Output	When isw2=1 z[0] : Value of z or t z[1] : Value of a normal distribution $z_{\frac{\alpha}{2}}$ or value of a t distribution $t_{\frac{\alpha}{2}}$ When isw2=2 z[0] : Value of z or t z[1] : Value of a normal distribution $-z_\alpha$ or value of a t distribution $-t_\alpha$ When isw2=3 z[0] : Value of z or t z[1] : Value of a normal distribution z_α or value of a t distribution t_α
8	isw1	I	1	Input	Switch for variance isw1=1 : The variance of the population is entered for xv isw1=2 : he variance (not an unbiased estimate) of the sample data is entered for xv isw1=3 : The variance (unbiased estimate) of the sample data is entered for xv

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
9	isw2	I	1	Input	The alternative hypothesis switch isw2=1 : When the alternative hypothesis is $\mu \neq \mu_0$ isw2=2 : When the alternative hypothesis is $\mu > \mu_0$ isw2=3 : When the alternative hypothesis is $\mu < \mu_0$
10	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) isw1 $\in \{1, 2, 3\}$

(b) isw2 $\in \{1, 2, 3\}$

(c) n ≥ 2

(d) xv > 0.0

(e) $0.0 \leq$ cl ≤ 100.0

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
1000	isw2=1 : cl=100.0 isw2=2 or 3 : cl=0.0 or cl=100.0.	isw2=1 : The positive maximum value is set for z[1]. isw2=2 or 3 : The positive maximum value or the negative minimum value is set for z[1].
3000	Restriction (a) was not satisfied.	Processing is aborted.
3010	Restriction (b) was not satisfied.	
3020	Restriction (c) was not satisfied.	
3030	Restriction (d) was not satisfied.	
3040	Restriction (e) was not satisfied.	

(6) **Notes**

None

(7) **Example**

(a) Problem

When the number of sample data is 10, the sample mean is 20.54 and the population variance is 0.08, test the hypothesis $\mu = 20.52$ related to the population mean μ with 95% confidence level. Assume that the alternative hypothesis is $\mu \neq 20.52$.

(b) Input data

isw1=1, isw2=1, n=10, xe=20.54, xv=0.08, xi=20.52 and cl=95.0.

(c) Main program

```
/*      C interface example for ASL_d3tsme */

#include <stdio.h>
#include <stdlib.h>
#include <asl.h>

int main()
{
    int n;
    double xe;
    double xv;
    double cl;
    double xi;
    int ir;
    double z[2];
    int isw1;
    int isw2;
    int ierr;
    int i;

    printf( "    *** ASL_d3tsme ***\n" );
    printf( "\n     ** Input **\n\n" );

    isw1=1;
    isw2=1;
    n=10;
    xe=20.54e0;
    xv=0.08e0;
    xi=20.52e0;
    cl=95.0e0;

    printf( "\tisw1= %6d\n", isw1);
    printf( "\tisw2= %6d\n", isw2);
    printf( "\tn   = %6d\n", n);
    printf( "\txe  = %8.3g\n", xe);
    printf( "\txv  = %8.3g\n", xv);
    printf( "\txi  = %8.3g\n", xi);
    printf( "\tcl  = %8.3g\n", cl);

    ierr = ASL_d3tsme(n, xe, xv, cl, xi, &ir, z, isw1, isw2);

    printf( "\n     ** Output **\n\n" );
    printf( "\tierr  = %4d\n\n", ierr );

    if ( ir == 0 ){
        printf( "\tHypothesis, mu = %8.3g, is accepted.\n\n", xi );
    }else{
        printf( "\tHypothesis, mu = %8.3g, is rejected.\n\n", xi );
    }

    for( i=0 ; i<2 ; i++ )
    {
        printf( "\tz[%2d] = %8.3g\n", i, z[i] );
    }

    return 0;
}
```

(d) Output results

```
*** ASL_d3tsme ***

 ** Input **

isw1=       1
isw2=       1
n   =      10
xe  =    20.5
xv  =    0.08
xi  =    20.5
cl  =      95

 ** Output **

ierr  =    0

Hypothesis, mu =     20.5, is accepted.

z[ 0] =    0.224
z[ 1] =     1.96
```

6.3.4 ASL_d3tssu, ASL_r3tssu
Test of the Difference of the Population Means According to Two Sets of Independent Samples

(1) **Function**

From the means μ_{x_1} and μ_{x_2} and variances (or population variances) σ_1^2 and σ_2^2 of two sets of independent sample data of sizes n_1 and n_2, respectively, test the hypothesis $\mu_1 = \mu_2$ related to the means μ_1 and μ_2 in the population to which the respective sets of sample data belong with the confidence level $1 - \alpha$. The test criteria are as follows.

(a) When the population variances are known

i. When the alternative hypothesis is $\mu_1 \neq \mu_2$ For z defined as follows

$$z = \frac{\mu_{x_1} - \mu_{x_2}}{\sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}}$$

$$\begin{cases} \text{If } |z| \geq z_{\frac{\alpha}{2}}, \text{ reject} \\ \text{If } |z| < z_{\frac{\alpha}{2}}, \text{ accept} \end{cases}$$

where $\frac{\alpha}{2}$ is as follows.

$$\frac{\alpha}{2} = 1 - P(z_{\frac{\alpha}{2}})$$

Here, $P(z)$ is the c.d.f. value of the standard normal distribution.

σ_1^2, σ_2^2 : Population variances of the two sets

ii. When the alternative hypothesis is $\mu_1 < \mu_2$

For z defined as follows

$$z = \frac{\mu_{x_1} - \mu_{x_2}}{\sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}}$$

$$\begin{cases} \text{If } z \leq -z_{\alpha}, \text{ reject} \\ \text{If } z > -z_{\alpha}, \text{ accept} \end{cases}$$

where α is as follows.

$$\alpha = 1 - P(z_{\alpha})$$

Here, $P(z)$ is the c.d.f. value of the standard normal distribution.

σ_1^2, σ_2^2 : Population variances of the two sets

iii. When the alternative hypothesis is $\mu_1 > \mu_2$

For z defined as follows

$$z = \frac{\mu_{x_1} - \mu_{x_2}}{\sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}}$$

$$\begin{cases} \text{If } z \geq z_{\alpha}, \text{ reject} \\ \text{If } z < z_{\alpha}, \text{ accept} \end{cases}$$

where α is as follows.

$$\alpha = 1 - P(z_{\alpha})$$

Here, $P(z)$ is the c.d.f. value of the standard normal distribution.

σ_1^2, σ_2^2 : Population variances of the two sets

(b) When the population variances of the two sets are equal and that value is unknown

i. When the alternative hypothesis is $\mu_1 \neq \mu_2$

For t defined as follows

$$t = \frac{\mu_{x_1} - \mu_{x_2}}{\sqrt{s_p^2(\frac{1}{n_1} + \frac{1}{n_2})}}$$

$$\begin{cases} \text{If } |t| \geq t_{\frac{\alpha}{2}}, \text{ reject} \\ \text{If } |t| < t_{\frac{\alpha}{2}}, \text{ accept} \end{cases}$$

where s_p^2 and $\frac{\alpha}{2}$ are as follows.

$$s_p^2 = \frac{(n_1-1)\sigma_1^2 + (n_2-1)\sigma_2^2}{n_1+n_2-2}$$

$$\frac{\alpha}{2} = 1 - P(t_{\frac{\alpha}{2}}|n_1+n_2-2)$$

Here, $P(t|n)$ is the cumulative distribution function (c.d.f.) of a t distribution with n degrees of freedom. σ_1^2, σ_2^2 : Unbiased estimates of the population variances.

ii. When the alternative hypothesis is $\mu_1 < \mu_2$

For t defined as follows

$$t = \frac{\mu_{x_1} - \mu_{x_2}}{\sqrt{s_p^2(\frac{1}{n_1}+\frac{1}{n_2})}}$$

$$\begin{cases} \text{If } t \leq -t_\alpha, \text{ reject} \\ \text{If } t > -t_\alpha, \text{ accept} \end{cases}$$

where s_p^2 and α are as follows.

$$s_p^2 = \frac{(n_1-1)\sigma_1^2 + (n_2-1)\sigma_2^2}{n_1+n_2-2}$$

$$\alpha = 1 - P(t_\alpha|n_1+n_2-2)$$

Here, $P(t|n)$ is the cumulative distribution function (c.d.f.) of a t distribution with n degrees of freedom.

σ_1^2, σ_2^2 : Unbiased estimates of the population variances.

iii. When the alternative hypothesis is $\mu_1 > \mu_2$

For t defined as follows

$$t = \frac{\mu_{x_1} - \mu_{x_2}}{\sqrt{s_p^2(\frac{1}{n_1}+\frac{1}{n_2})}}$$

$$\begin{cases} \text{If } t \geq t_\alpha, \text{ reject} \\ \text{If } t < t_\alpha, \text{ accept} \end{cases}$$

where s_p^2 and α are as follows.

$$s_p^2 = \frac{(n_1-1)\sigma_1^2 + (n_2-1)\sigma_2^2}{n_1+n_2-2}$$

$$\alpha = 1 - P(t_\alpha|n_1+n_2-2)$$

Here, $P(t|n)$ is the cumulative distribution function (c.d.f.) of a t distribution with n degrees of freedom.

σ_1^2, σ_2^2 : Unbiased estimates of the population variances.

(c) When the population variances of the two sets are not equal and those values are unknown

i. When the alternative hypothesis is $\mu_1 \neq \mu_2$

For t defined as follows

$$t = \frac{\mu_{x_1} - \mu_{x_2}}{\sqrt{\frac{\sigma_1^2}{n_1}+\frac{\sigma_2^2}{n_2}}}$$

$$t_{\frac{\alpha}{2}}^* = \frac{\beta_1 t_{\frac{\alpha}{2}}^{(1)} + \beta_2 t_{\frac{\alpha}{2}}^{(2)}}{\beta_1+\beta_2}$$

$$\begin{cases} \text{If } |t| \geq t_{\frac{\alpha}{2}}^*, \text{ reject} \\ \text{If } |t| < t_{\frac{\alpha}{2}}^*, \text{ accept} \end{cases}$$

where $\frac{\alpha}{2}$ is as follows.

$$\frac{\alpha}{2} = 1 - P(t_{\frac{\alpha}{2}}^{(1)}|n_1-1) = 1 - P(t_{\frac{\alpha}{2}}^{(2)}|n_2-1)$$

Here, $P(t|n)$ is the cumulative distribution function (c.d.f.) of a t distribution with n degrees of freedom.

$$\beta_1 = \frac{\sigma_1^2}{n_1}, \beta_2 = \frac{\sigma_2^2}{n_2}$$

σ_1^2, σ_2^2 : Unbiased estimates of the population variances

ii. When the alternative hypothesis is $\mu_1 < \mu_2$

For t defined as follows

$$t = \frac{\mu_{x_1} - \mu_{x_2}}{\sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}}$$

$$t_\alpha^* = \frac{\beta_1 t_\alpha^{(1)} + \beta_2 t_\alpha^{(2)}}{\beta_1 + \beta_2}$$

$$\begin{cases} \text{If } t \leq -t_\alpha^*, \text{ reject} \\ \text{If } t > -t_\alpha^*, \text{ accept} \end{cases}$$

where α is as follows.

$$\alpha = 1 - P(t_\alpha^{(1)}|n_1 - 1) = 1 - P(t_\alpha^{(2)}|n_2 - 1)$$

Here, $P(t|n)$ is the cumulative distribution function (c.d.f.) of a t distribution with n degrees of freedom.

$$\beta_1 = \frac{\sigma_1^2}{n_1}, \beta_2 = \frac{\sigma_2^2}{n_2}$$

σ_1^2, σ_2^2 : Unbiased estimates of the population variances

iii. When the alternative hypothesis is $\mu_1 > \mu_2$

For t defined as follows

$$t = \frac{\mu_{x_1} - \mu_{x_2}}{\sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}}$$

$$t_\alpha^* = \frac{\beta_1 t_\alpha^{(1)} + \beta_2 t_\alpha^{(2)}}{\beta_1 + \beta_2}$$

$$\begin{cases} \text{If } t \geq t_\alpha^*, \text{ reject} \\ \text{If } t < t_\alpha^*, \text{ accept} \end{cases}$$

where α is as follows.

$$\alpha = 1 - P(t_\alpha^{(1)}|n_1 - 1) = 1 - P(t_\alpha^{(2)}|n_2 - 1)$$

Here, $P(t|n)$ is the cumulative distribution function (c.d.f.) of a t distribution with n degrees of freedom.

$$\beta_1 = \frac{\sigma_1^2}{n_1}, \beta_2 = \frac{\sigma_2^2}{n_2}$$

σ_1^2, σ_2^2 : Unbiased estimates of the population variances

(2) **Usage**

Double precision:

```
ierr = ASL_d3tssu  (n1, xe1, xv1, n2, xe2, xv2, cl, &ir, z, isw1, isw2);
```

Single precision:

```
ierr = ASL_r3tssu  (n1, xe1, xv1, n2, xe2, xv2, cl, &ir, z, isw1, isw2);
```

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	n1	I	1	Input	Number of sample data n_1
2	xe1	{D / R}	1	Input	Mean of sample data μ_{x_1}
3	xv1	{D / R}	1	Input	Variance of sample data or population ${\sigma_1}^2$
4	n2	I	1	Input	Number of sample data n_2
5	xe2	{D / R}	1	Input	Mean of sample data μ_{x_2}
6	xv2	{D / R}	1	Input	Variance of sample data or population ${\sigma_2}^2$
7	cl	{D / R}	1	Input	Confidence level $100(1-\alpha)(\%)$
8	ir	I*	1	Output	Test result ir=0 : Hypothesis $\mu_1 = \mu_2$ is accepted ir=1 : Hypothesis $\mu_1 = \mu_2$ is rejected
9	z	{D* / R*}	2	Output	When isw2=1 z[0] : Value of z or t z[1] : Value of a standard normal distribution $z_{\frac{\alpha}{2}}$ or value of t distribution $t_{\frac{\alpha}{2}}$ or $t^*_{\frac{\alpha}{2}}$ When isw2=2 z[0] : Value of z or t z[1] : Value of a standard normal distribution $-z_\alpha$ or value of t distribution $-t_\alpha$ or $-t^*_\alpha$ When isw2=3 z[0] : Value of z or t z[1] : Value of a standard normal distribution z_α or value of t distribution t_α or t^*_α

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
10	isw1	I	1	Input	Switch for variance isw1=1 : The population variances of the two sets are entered for xv1 and xv2 isw1=2 : The population variances of the two sets are equal, and the variances (not unbiased estimates) of the sample data are entered for xv1 and xv2 isw1=3 : The population variances of the two sets are equal, and the variances (unbiased estimates) of the sample data are entered for xv1 and xv2 isw1=4 : The population variances of the two sets are not equal, and the variances (not unbiased estimates) of the sample data are entered for xv1 and xv2 isw1=5 : The population variances of the two sets are not equal, and the variances (unbiased estimates) of the sample data are entered for xv1 and xv2
11	isw2	I	1	Input	The alternative hypothesis switch isw2=1 : When the alternative hypothesis is $\mu_1 \neq \mu_2$ isw2=2 : When the alternative hypothesis is $\mu_1 < \mu_2$ isw2=3 : When the alternative hypothesis is $\mu_1 > \mu_2$
12	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) isw1 $\in \{1, 2, 3, 4, 5\}$

(b) isw2 $\in \{1, 2, 3\}$

(c) n1$\geq$ 2, n2$\geq$ 2

(d) xv1$\geq$ 0.0, xv2$\geq$ 0.0

(e) $0.0 \leq$ cl ≤ 100.0

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
1000	isw2=1 : cl=100.0 isw2=2 or 3 : cl=0.0 or cl=100.0.	isw2=1 : The positive maximum value is set for z[1]. isw2=2 or 3 : The positive maximum value or the negative minimum value is set for z[1].
3000	Restriction (a) was not satisfied.	Processing is aborted.
3010	Restriction (b) was not satisfied.	
3020	Restriction (c) was not satisfied.	
3030	Restriction (d) was not satisfied.	
3040	Restriction (e) was not satisfied.	

(6) **Notes**

None

(7) **Example**

(a) Problem

When the number of sample data, the sample mean and the population variance in the two sets of independent samples are 20, 62.0, 64.0 and 25, 67.0, 81.0, respectively, test the hypothesis $\mu_1 = \mu_2$ related to the each population mean μ_1 and μ_2 with 95% confidence level. Assume that the alternative hypothesis is $\mu_1 \neq \mu_2$.

(b) Input data

isw1=1, isw2=1, n1=20, xe1=62.0, xv1=64.0, n2=25, xe2=67.0, xv2=81.0 and cl=95.0.

(c) Main program

```
/*      C interface example for ASL_d3tssu */

#include <stdio.h>
#include <stdlib.h>
#include <asl.h>

int main()
{
    int n1;
    double xe1;
    double xv1;
    int n2;
    double xe2;
    double xv2;
    double cl;
    int ir;
    double z[2];
    int isw1;
    int isw2;
    int ierr;
    int i;


    printf( "    *** ASL_d3tssu ***\n" );
    printf( "\n     ** Input **\n\n" );

    isw1=1;
    isw2=1;
    n1=20;
    xe1=62.0e0;
    xv1=64.0e0;
    n2=25;
    xe2=67.0e0;
    xv2=81.0e0;
    cl=95.0e0;
```

```
    printf( "\tisw1= %6d\n", isw1);
    printf( "\tisw2= %6d\n", isw2);
    printf( "\tn1  = %6d\n", n1);
    printf( "\txe1 = %8.3g\n", xe1);
    printf( "\txv1 = %8.3g\n", xv1);
    printf( "\tn2  = %6d\n", n2);
    printf( "\txe2 = %8.3g\n", xe2);
    printf( "\txv2 = %8.3g\n", xv2);
    printf( "\tcl  = %8.3g\n", cl);

    ierr = ASL_d3tssu(n1, xe1, xv1, n2, xe2, xv2, cl, &ir, z, isw1, isw2);

    printf( "\n     ** Output **\n\n" );
    printf( "\tierr  = %4d\n\n", ierr );

    if ( ir == 0 ){
        printf( "\tHypothesis, mu_1 = mu_2, is accepted.\n\n");
    }else{
        printf( "\tHypothesis, mu_1 = mu_2, is rejected.\n\n");
    }

    for( i=0 ; i<2 ; i++ )
    {
        printf( "\tz[%2d] = %8.3g\n", i, z[i] );
    }

    return 0;
}
```

(d) Output results

```
*** ASL_d3tssu ***

 ** Input **

isw1=      1
isw2=      1
n1  =     20
xe1 =       62
xv1 =       64
n2  =     25
xe2 =       67
xv2 =       81
cl  =       95

 ** Output **

ierr  =    0

Hypothesis, mu_1 = mu_2, is rejected.

z[ 0] =    -1.97
z[ 1] =     1.96
```

6.3.5 ASL_d3tsva, ASL_r3tsva
Test of the Population Variance Due to One Set of Samples

(1) **Function**

From the variance (or population variance) s^2 of one set of sample data of size n, test the hypothesis $\sigma^2 = \sigma_0^2$ related to the population variance σ^2 in the population to which the respective set of sample data belong with the confidence level $1 - \alpha$. The test criteria are as follows.

(a) When the alternative hypothesis is $\sigma^2 \neq \sigma_0^2$

For χ^2 defined as follows

$$\chi^2 = \frac{(n-1)s^2}{\sigma_0^2}$$

$$\begin{cases} \text{If } \chi^2 \leq \chi^2_{1-\frac{\alpha}{2}} \text{ or } \chi^2 \geq \chi^2_{\frac{\alpha}{2}}, \text{ reject} \\ \text{If } \chi^2_{1-\frac{\alpha}{2}} < \chi^2 < \chi^2_{\frac{\alpha}{2}}, \text{ accept} \end{cases}$$

where $\frac{\alpha}{2}$ is as follows.

$$\frac{\alpha}{2} = 1 - P(\chi^2_{\frac{\alpha}{2}}|n-1) = P(\chi^2_{1-\frac{\alpha}{2}}|n-1)$$

$P(\chi^2|n)$ is the cumulative distribution function (c.d.f.) of a χ^2 distribution having number of degrees of freedom n.

s^2 : Unbiased estimate of population variance

(b) When the alternative hypothesis is $\sigma^2 < \sigma_0^2$

For χ^2 defined as follows

$$\chi^2 = \frac{(n-1)s^2}{\sigma_0^2}$$

$$\begin{cases} \text{If } \chi^2 \leq \chi^2_{1-\alpha}, \text{ reject} \\ \text{If } \chi^2 > \chi^2_{1-\alpha}, \text{ accept} \end{cases}$$

where α is as follows.

$$\alpha = P(\chi^2_{1-\alpha}|n-1)$$

$P(\chi^2|n)$ is the cumulative distribution function (c.d.f.) of a χ^2 distribution having number of degrees of freedom n.

s^2 : Unbiased estimate of population variance

(c) When the alternative hypothesis is $\sigma^2 > \sigma_0^2$

For χ^2 defined as follows

$$\chi^2 = \frac{(n-1)s^2}{\sigma_0^2}$$

$$\begin{cases} \text{If } \chi^2 \geq \chi^2_{\alpha}, \text{ reject} \\ \text{If } \chi^2 < \chi^2_{\alpha}, \text{ accept} \end{cases}$$

where α is as follows.

$$\alpha = P(\chi^2_{\alpha}|n-1)$$

$P(\chi^2|n)$ is the cumulative distribution function (c.d.f.) of a χ^2 distribution having number of degrees of freedom n.

s^2 : Unbiased estimate of population variance

(2) **Usage**

Double precision:

ierr = ASL_d3tsva (n, xv, cl, xi, &ir, z, isw1, isw2);

Single precision:

ierr = ASL_r3tsva (n, xv, cl, xi, &ir, z, isw1, isw2);

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	n	I	1	Input	Number of sample data n
2	xv	{D / R}	1	Input	Variance of sample data s^2
3	cl	{D / R}	1	Input	Confidence level $100(1-\alpha)(\%)$
4	xi	{D / R}	1	Input	Tested variance σ_0
5	ir	I*	1	Output	Test result ir=0 : Hypothesis $\sigma^2 = \sigma_0^2$ is accepted ir=1 : Hypothesis $\sigma^2 = \sigma_0^2$ is rejected
6	z	{D* / R*}	3	Output	z[0] : Value of χ^2 z[1] : Value of χ^2 distribution $\chi^2_{\frac{\alpha}{2}}$ z[2] : Value of χ^2 distribution $\chi^2_{1-\frac{\alpha}{2}}$
7	isw1	I	1	Input	isw1=1 : The variance (not an unbiased estimate) of the sample data is entered for xv isw1=2 : The variance (unbiased estimate) of the sample data is entered for xv
8	isw2	I	1	Input	The alternative hypothesis switch isw2=1 : When the alternative hypothesis is $\sigma^2 \neq \sigma_0^2$ isw2=2 : When the alternative hypothesis is $\sigma^2 < \sigma_0^2$ isw2=3 : When the alternative hypothesis is $\sigma^2 > \sigma_0^2$
9	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) isw1 $\in \{1, 2\}$

(b) isw2 $\in \{1, 2, 3\}$

(c) n ≥ 2

(d) xv > 0.0

(e) xi > 0.0

(f) $0.0 \leq$ cl ≤ 100.0

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
1000	isw2=1 : cl=100.0 isw2=2 or 3 : cl=0.0 or cl=100.0.	isw2=1 : The positive maximum value is set for z[1]. isw2=2 or 3 : 0.0 or the positive maximum value is set for z[1].
3000	Restriction (a) was not satisfied.	Processing is aborted.
3010	Restriction (b) was not satisfied.	
3020	Restriction (c) was not satisfied.	
3030	Restriction (d) was not satisfied.	
3040	Restriction (e) was not satisfied.	

(6) **Notes**

None

(7) **Example**

(a) Problem

When the number of sample data is 25 and the sample variance (unbiased estimate) is 182.25, test the hypothesis $\sigma^2 = 100.0$ related to the population variance with 95% confidence level. Assume that the alternative hypothesis is $\sigma^2 \neq 100.0$.

(b) Input data

isw1=2, isw2=1, n=25, xv=182.25, xi=100.0 and cl=95.0.

(c) Main program

```
/*      C interface example for ASL_d3tsva */

#include <stdio.h>
#include <stdlib.h>
#include <asl.h>

int main()
{
    int n;
    double xv;
    double cl;
    double xi;
    int ir;
    double z[3];
    int isw1;
    int isw2;
    int ierr;
    int i;
```

```
    printf( "    *** ASL_d3tsva ***\n" );
    printf( "\n     ** Input **\n\n" );

    isw1=2;
    isw2=1;
    n=25;
    xv=182.25e0;
    xi=100.0e0;
    cl=95.0e0;

    printf( "\tisw1= %6d\n", isw1);
    printf( "\tisw2= %6d\n", isw2);
    printf( "\tn   = %6d\n", n);
    printf( "\txv  = %10.6g\n", xv);
    printf( "\txi  = %10.6g\n", xi);
    printf( "\tcl  = %10.6g\n", cl);

    ierr = ASL_d3tsva(n, xv, cl, xi, &ir, z, isw1, isw2);

    printf( "\n     ** Output **\n\n" );
    printf( "\tierr  = %6d\n\n", ierr );

    if ( ir == 0 ){
        printf( "\tHypothesis, sigma2 = %8.3g, is accepted.\n\n", xi );
    }else{
        printf( "\tHypothesis, sigma2 = %8.3g, is rejected.\n\n", xi );
    }

    for( i=0 ; i<3 ; i++ )
    {
        printf( "\tz[%2d] = %8.3g\n", i, z[i] );
    }

    return 0;
}
```

(d) Output results

```
*** ASL_d3tsva ***

 ** Input **

isw1=      2
isw2=      1
n   =     25
xv  =     182.25
xi  =        100
cl  =         95

 ** Output **

ierr  =      0

Hypothesis, sigma2 =      100, is rejected.

z[ 0] =     43.7
z[ 1] =     39.4
z[ 2] =     12.4
```

6.3.6 ASL_d3tstc, ASL_r3tstc
Test of the Population Correlation Coefficient According to One Set of Samples

(1) **Function**

From the sample correlation coefficient r of one set of sample data of size n, test the hypothesis $\rho = \rho_0$ related to the population correlation coefficient ρ in the population to which the respective set of sample data belong with the confidence level $1 - \alpha$. The test criteria are as follows.

(a) Hypothesis: $\rho = 0$

i. When the alternative hypothesis is $\rho \neq 0$

For t defined as follows

$$t = r\sqrt{\frac{n-2}{1-r^2}}$$

$$\begin{cases} \text{If } |t| \geq t_{\frac{\alpha}{2}}, \text{ reject} \\ \text{If } |t| < t_{\frac{\alpha}{2}}, \text{ accept} \end{cases}$$

where, $\frac{\alpha}{2}$ is as follows.

$$\frac{\alpha}{2} = 1 - P(t_{\frac{\alpha}{2}}|n-2)$$

Here, $P(t|n)$ is the cumulative distribution function (c.d.f.) of a t distribution with n degrees of freedom.

ii. When the alternative hypothesis is $\rho < 0$

For t defined as follows

$$t = r\sqrt{\frac{n-2}{1-r^2}}$$

$$\begin{cases} \text{If } t \geq -t_{\alpha}, \text{ reject} \\ \text{If } t < -t_{\alpha}, \text{ accept} \end{cases}$$

where, $\frac{\alpha}{2}$ is as follows.

$$\frac{\alpha}{2} = 1 - P(t_{\alpha}|n-2)$$

Here, $P(t|n)$ is the cumulative distribution function (c.d.f.) of a t distribution with n degrees of freedom.

iii. When the alternative hypothesis is $\rho > 0$

For t defined as follows

$$t = r\sqrt{\frac{n-2}{1-r^2}}$$

$$\begin{cases} \text{If } t \geq t_{\alpha}, \text{ reject} \\ \text{If } t < t_{\alpha}, \text{ accept} \end{cases}$$

where, $\frac{\alpha}{2}$ is as follows.

$$\frac{\alpha}{2} = 1 - P(t_{\alpha}|n-2)$$

Here, $P(t|n)$ is the cumulative distribution function (c.d.f.) of a t distribution with n degrees of freedom.

(b) Hypothesis: $\rho = \rho_0$

i. When the alternative hypothesis is $\rho \neq \rho_0$

For t defined as follows

$$t = (z - z_0)\sqrt{n-3}$$

$$\begin{cases} \text{If } |t| \geq z_{\frac{\alpha}{2}}, \text{ reject} \\ \text{If } |t| < z_{\frac{\alpha}{2}}, \text{ accept} \end{cases}$$

where, z, z_0 and $\frac{\alpha}{2}$ are as follows.

$$z = \frac{1}{2}\log_e \frac{1+r}{1-r}$$
$$z_0 = \frac{1}{2}\log_e \frac{1+\rho_0}{1-\rho_0}$$
$$\frac{\alpha}{2} = 1 - P(z_{\frac{\alpha}{2}})$$

Here, $P(z)$ is the c.d.f. value of the standard normal distribution.

ii. When the alternative hypothesis is $\rho < \rho_0$

For t defined as follows

$$t = (z - z_0)\sqrt{n-3}$$

$$\begin{cases} \text{If } t \leq -z_\alpha, \text{ reject} \\ \text{If } t > -z_\alpha, \text{ accept} \end{cases}$$

where, z, z_0 and $\frac{\alpha}{2}$ are as follows.

$$z = \frac{1}{2}\log_e \frac{1+r}{1-r}$$
$$z_0 = \frac{1}{2}\log_e \frac{1+\rho_0}{1-\rho_0}$$
$$\frac{\alpha}{2} = 1 - P(z_\alpha)$$

Here, $P(z)$ is the c.d.f. value of the standard normal distribution.

iii. When the alternative hypothesis is $\rho > \rho_0$

For t defined as follows

$$t = (z - z_0)\sqrt{n-3}$$

$$\begin{cases} \text{If } t \geq z_\alpha, \text{ reject} \\ \text{If } t < z_\alpha, \text{ accept} \end{cases}$$

where, z, z_0 and $\frac{\alpha}{2}$ are as follows.

$$z = \frac{1}{2}\log_e \frac{1+r}{1-r}$$
$$z_0 = \frac{1}{2}\log_e \frac{1+\rho_0}{1-\rho_0}$$
$$\frac{\alpha}{2} = 1 - P(z_\alpha)$$

Here, $P(z)$ is the c.d.f. value of the standard normal distribution.

(2) **Usage**

Double precision:

```
ierr = ASL_d3tstc  (n, r, cl, r0, &ir, z, isw);
```

Single precision:

```
ierr = ASL_r3tstc  (n, r, cl, r0, &ir, z, isw);
```

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	n	I	1	Input	Number of sample data n
2	r	{D / R}	1	Input	The sample correlation coefficient r of one set of sample data (See Note (a))
3	cl	{D / R}	1	Input	Confidence level $100(1-\alpha)$
4	r0	{D / R}	1	Input	Value of tested correlation coefficient ρ_0
5	ir	I*	1	Output	Test result ir=0 : Hypothesis $\rho = \rho_0$ is accepted ir=1 : Hypothesis $\rho = \rho_0$ is rejected
6	z	{D*/ R*}	2	Output	When isw=1 z[0] : Value of t z[1] : Value of t distribution $t_{\frac{\alpha}{2}}$ or value of a standard normal distribution $z_{\frac{\alpha}{2}}$ When isw=2 z[0] : Value of t z[1] : Value of t distribution $-t_\alpha$ or value of a standard normal distribution $-z_\alpha$
7	isw	I	1	Input	The alternative hypothesis switch isw=1 : When the alternative hypothesis is $\rho \neq \rho_0$ isw=2 : When the alternative hypothesis is $\rho < \rho_0$ isw=3 : When the alternative hypothesis is $\rho > \rho_0$
8	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) isw $\in \{1, 2, 3\}$

(b) n$\geq$4

(c) $-1.0 <$ r < 1.0

(d) $-1.0 <$ r0 < 1.0

(e) $0.0 \leq$ cl ≤ 100.0

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
1000	isw=1 : cl=100.0 isw=2 or 3 : cl=0.0 or cl=100.0.	isw=1 : The positive maximum value is set for z[1]. isw=2 or 3 : The positive maximum value or the negative minimum value is set for z[1].
3000	Restriction (a) was not satisfied.	Processing is aborted.
3010	Restriction (b) was not satisfied.	
3020	Restriction (c) was not satisfied.	
3030	Restriction (d) was not satisfied.	
3040	Restriction (e) was not satisfied.	

(6) **Notes**

(a) The sample correlation coefficient r for n sample data, $\{x_i, y_i\}$ $(i = 1, \cdots, n)$, is defined as follows. (See 4.4.1 $\left\{\begin{matrix}\text{ASL\_d2ccmt}\\ \text{ASL\_r2ccmt}\end{matrix}\right\}$)

$$r = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu_x)(y_i - \mu_y)}{\sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu_x)^2}\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \mu_y)^2}}$$

(7) **Example**

(a) Problem

From the following one set of sample data of size 10, test the hypothesis $\rho = 0.0$ related to the population correlation coefficient ρ with 95% confidence level.

x_i	y_i
10.129	63.4
12.611	60.1
13.900	57.2
16.532	46.5
20.822	43.9
26.025	39.6
28.283	39.7
29.199	39.1
30.766	37.8
32.664	27.8

Assume that the alternative hypothesis is $\rho \neq 0.0$.

(b) Input data

n=10, r0=0.0 and cl=95.0.

(c) Main program

```
/*      C interface example for ASL_d3tstc */

#include <stdio.h>
#include <stdlib.h>
#include <asl.h>

int main()
{
    int n;
    double r;
    double cl;
    double r0;
    int ir;
    double z[2];
    int isw_tstc;
    int ierr;
    double *a;
    int na;
    int m;
    int nr;
    int ns;
    double *x1;
    double *rr;
    double *wk;
    int isw_ccmt;
    int kerr;
    int i, j;
    FILE *fp;

    fp = fopen( "d3tstc.dat", "r" );
    if( fp == NULL )
    {
        printf( "file open error\n" );
        return -1;
    }

    printf( "    *** ASL_d3tstc ***\n" );
    printf( "\n     ** Input **\n\n" );

    fscanf( fp, "%lf", &cl );
    fscanf( fp, "%lf", &r0 );
    fscanf( fp, "%d", &isw_tstc );
    fscanf( fp, "%d", &na );
    fscanf( fp, "%d", &n );
    fscanf( fp, "%d", &m );
    fscanf( fp, "%d", &nr );
    fscanf( fp, "%d", &isw_ccmt );

    a = ( double * )malloc((size_t)( sizeof(double) * (na*m) ));
    if( a == NULL )
    {
        printf( "no enough memory for array a\n" );
        return -1;
    }

    x1 = ( double * )malloc((size_t)( sizeof(double) * m ));
    if( x1 == NULL )
    {
        printf( "no enough memory for array x1\n" );
        return -1;
    }

    rr = ( double * )malloc((size_t)( sizeof(double) * (nr*m) ));
    if( rr == NULL )
    {
        printf( "no enough memory for array rr\n" );
        return -1;
    }

    wk = ( double * )malloc((size_t)( sizeof(double) * m ));
    if( wk == NULL )
    {
        printf( "no enough memory for array wk\n" );
        return -1;
    }

    printf( "\t**  ASL_d2ccmt  **\n");
    printf( "\tna       = %6d\n", na);
    printf( "\tn        = %6d\n", n);
    printf( "\tm        = %6d\n", m);
    printf( "\tnr       = %6d\n", nr);
    printf( "\tisw_ccmt = %6d\n", isw_ccmt);

    printf("\n\t sample1 sample2\n");
    for( i=0; i<n; i++ )
    {
        printf("\t");
        for( j=0; j<m; j++ )
        {
```

```
            fscanf( fp, "%lf", &a[i+na*j] );
            printf( "%8.3g", a[i+na*j] );
        }
        printf("\n");
    }
    fclose( fp );

    kerr = ASL_d2ccmt(a, na, n, m, &ns, x1, rr, nr, isw_ccmt, wk);

    if( kerr != 0 )
    {
        printf("Error occured in ASL_d2ccmt. kerr=%6d\n", kerr);
        return -1;
    }

    r=rr[1];

    printf( "\n\t**  ASL_d3tstc  **\n");
    printf( "\tn       = %6d\n", n);
    printf( "\tr       = %8.3g\n", r);
    printf( "\tr0      = %8.3g\n", r0);
    printf( "\tcl      = %8.3g\n", cl);
    printf( "\tisw_tstc= %6d\n", isw_tstc);

    ierr = ASL_d3tstc(n, r, cl, r0, &ir, z, isw_tstc);

    printf( "\n     ** Output **\n\n" );
    printf( "\t** ASL_d2ccmt **\n");
    printf( "\tkerr = %6d\n", kerr );
    printf( "\tr    = %8.3g\n", r );
    printf( "\n\t** ASL_d3tstc **\n");
    printf( "\tierr = %6d\n\n", ierr );

    if ( ir == 0 ){
        printf( "\tHypothesis, rho = %8.3g, is accepted.\n\n", r0 );
    }else{
        printf( "\tHypothesis, rho = %8.3g, is rejected.\n\n", r0 );
    }

    for( i=0 ; i<2 ; i++ )
    {
        printf( "\tz[%2d] = %8.3g\n", i, z[i] );
    }

    free( a );
    free( rr );
    free( x1 );
    free( wk );

    return 0;
}
```

(d) Output results

```
*** ASL_d3tstc ***

 ** Input **

**  ASL_d2ccmt  **
na       =     10
n        =     10
m        =      2
nr       =     10
isw_ccmt =      0

 sample1 sample2
    10.1    63.4
    12.6    60.1
    13.9    57.2
    16.5    46.5
    20.8    43.9
      26    39.6
    28.3    39.7
    29.2    39.1
    30.8    37.8
    32.7    37.8

**  ASL_d3tstc  **
n        =     10
r        =   -0.945
r0       =      0
cl       =     95
isw_tstc=      1

 ** Output **

** ASL_d2ccmt **
kerr =      0
r    =   -0.945

** ASL_d3tstc **
```

```
ierr =      0

Hypothesis, rho =        0, is rejected.

z[ 0] =    -8.15
z[ 1] =     2.31
```

6.3.7 ASL_d3tscd, ASL_r3tscd
Test of the Difference of the Population Correlation Coefficients According to Two Sets of Independent Samples

(1) **Function**

From the correlation coefficients r_1 and r_2 of two sets of independent sample data of sizes n_1 and n_2, respectively, test the hypothesis $\rho_1 = \rho_2$ related to the population correlation coefficients ρ_1 and ρ_2 in the population to which the respective set of sample data belong with the confidence level $1 - \alpha$. The test criteria are as follows.

(a) When the alternative hypothesis is $\rho_1 \neq \rho_2$

For t defined as follows

$$t = \frac{z_1 - z_2}{\sqrt{\frac{1}{n_1 - 3} + \frac{1}{n_2 - 3}}}$$

$$\begin{cases} \text{If } |t| \geq z_{\frac{\alpha}{2}}, \text{ reject} \\ \text{If } |t| < z_{\frac{\alpha}{2}}, \text{ accept} \end{cases}$$

where, z_1, z_2 and $\frac{\alpha}{2}$ are as follows.

$$z_1 = \frac{1}{2} \log_e \frac{1 + r_1}{1 - r_1}$$

$$z_2 = \frac{1}{2} \log_e \frac{1 + r_2}{1 - r_2}$$

$$\frac{\alpha}{2} = 1 - P(z_{\frac{\alpha}{2}})$$

Here, $P(z)$ is the c.d.f. value of the standard normal distribution.

(b) When the alternative hypothesis is $\rho_1 < \rho_2$

For t defined as follows

$$t = \frac{z_1 - z_2}{\sqrt{\frac{1}{n_1 - 3} + \frac{1}{n_2 - 3}}}$$

$$\begin{cases} \text{If } t \leq -z_{\alpha}, \text{ reject} \\ \text{If } t > -z_{\alpha}, \text{ accept} \end{cases}$$

where, z_1, z_2 and α are as follows.

$$z_1 = \frac{1}{2} \log_e \frac{1 + r_1}{1 - r_1}$$

$$z_2 = \frac{1}{2} \log_e \frac{1 + r_2}{1 - r_2}$$

$$\alpha = 1 - P(z_{\alpha})$$

Here, $P(z)$ is the c.d.f. value of the standard normal distribution.

(c) When the alternative hypothesis is $\rho_1 > \rho_2$

For t defined as follows

$$t = \frac{z_1 - z_2}{\sqrt{\frac{1}{n_1 - 3} + \frac{1}{n_2 - 3}}}$$

$$\begin{cases} \text{If } t \geq z_\alpha, \text{ reject} \\ \text{If } t < z_\alpha, \text{ accept} \end{cases}$$

where, z_1, z_2 and α are as follows.

$$z_1 = \frac{1}{2}\log_e \frac{1+r_1}{1-r_1}$$

$$z_2 = \frac{1}{2}\log_e \frac{1+r_2}{1-r_2}$$

$$\alpha = 1 - P(z_\alpha)$$

Here, $P(z)$ is the c.d.f. value of the standard normal distribution.

(2) **Usage**

Double precision:

```
ierr = ASL_d3tscd  (n1, r1, n2, r2, cl, &ir, z, isw);
```

Single precision:

```
ierr = ASL_r3tscd  (n1, r1, n2, r2, cl, &ir, z, isw);
```

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	n1	I	1	Input	Number of first sample data n_1
2	r1	{D / R}	1	Input	The correlation coefficient r_1 of first sample data (See Note (a))
3	n2	I	1	Input	Number of second sample data n_2
4	r2	{D / R}	1	Input	The correlation coefficient r_2 of second sample data (See Note (a))
5	cl	{D / R}	1	Input	Confidence level $100(1-\alpha)$
6	ir	I*	1	Output	Test result ir=0 : Hypothesis $\rho_1 = \rho_2$ is accepted ir=1 : Hypothesis $\rho_1 = \rho_2$ is rejected
7	z	{D* / R*}	2	Output	When isw=1 z[0] : Value of t z[1] : Value of a standard normal distribution $z_{\frac{\alpha}{2}}$ When isw=2 z[0] : Value of t z[1] : Value of a standard normal distribution $-z_\alpha$ When isw=3 z[0] : Value of t z[1] : Value of a standard normal distribution z_α

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
8	isw	I	1	Input	The alternative hypothesis switch isw=1 : When the alternative hypothesis is $\rho_1 \neq \rho_2$ isw=2 : When the alternative hypothesis is $\rho_1 < \rho_2$ isw=3 : When the alternative hypothesis is $\rho_1 > \rho_2$
9	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) isw $\in \{1, 2, 3\}$

(b) n1 ≥ 4, n2 ≥ 4

(c) $-1.0 <$ r1 < 1.0, $-1.0 <$ r2 < 1.0

(d) $0.0 \leq$ cl ≤ 100.0

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
1000	isw=1 : cl=100.0 isw=2 or 3 : cl=0.0 or cl=100.0.	isw=1 : The positive maximum value is set for z[1]. isw=2 or 3 : The positive maximum value or the negative minimum value is set for z[1].
3000	Restriction (a) was not satisfied.	Processing is aborted.
3010	Restriction (b) was not satisfied.	
3020	Restriction (c) was not satisfied.	
3030	Restriction (d) was not satisfied.	

(6) **Notes**

(a) The sample correlation coefficient r for n sample data, $\{x_i, y_i\}$ $(i = 1, \cdots, n)$, is defined as follows. (See 4.4.1 $\left\{\begin{matrix}\text{ASL\_d2ccmt}\\ \text{ASL\_r2ccmt}\end{matrix}\right\}$)

$$r = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu_x)(y_i - \mu_y)}{\sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu_x)^2}\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \mu_y)^2}}$$

(7) **Example**

See the example in Section 6.2.6 (7).

6.3.8 ASL_d3tssr, ASL_r3tssr
Test in the Simple Linear Regression

(1) **Function**

For regression coefficient a and constant term b in the following simple linear regression expression (or regression line) related to one set of sample data $\{x_i, y_i\}$ $(1, \cdots, n)$ of size n

$$\hat{y}_i = ax_i + b$$

test the hypothesis related to the regression coefficient A and constant term B in the population to which the respective set of sample data belong with the confidence level $1 - \alpha$. Assume that y_i corresponding to each x_i is the random sample from the normal population having the mean $Ax_i - B$ and the variance σ^2. Obtain the regression coefficient a and constant term b of the sample data from the following normal equations.

$$\begin{cases} \sum_{i=1}^{n} y_i = a\sum_{i=1}^{n} x_i + bn \\ \sum_{i=1}^{n} x_i y_i = a\sum_{i=1}^{n} x_i^2 + b\sum_{i=1}^{n} x_i \end{cases}$$

The test criteria are as follows.

(a) Regression coefficient

Hypothesis: $A = A_0$

i. When the population variance is known

A. When the alternative hypothesis is $A \neq A_0$

For t defined as follows

$$t = \frac{a - A_0}{s_a}$$

$$\begin{cases} \text{If } |t| \geq z_{\frac{\alpha}{2}}, \text{ reject} \\ \text{If } |t| < z_{\frac{\alpha}{2}}, \text{ accept} \end{cases}$$

where s_a is as follows.

$$s_a = \sqrt{\frac{\sigma^2}{\sum (x_i - \mu_x)^2}}$$

σ^2 : Population variance

$$\frac{\alpha}{2} = 1 - P(z_{\frac{\alpha}{2}})$$

Here, $P(z)$ is the c.d.f. value of the standard normal distribution.

B. When the alternative hypothesis is $A < A_0$

For t defined as follows

$$t = \frac{a - A_0}{s_a}$$

$$\begin{cases} \text{If } t \geq -z_{\alpha}, \text{ reject} \\ \text{If } t < -z_{\alpha}, \text{ accept} \end{cases}$$

where s_a is as follows.

$$s_a = \sqrt{\frac{\sigma^2}{\sum (x_i - \mu_x)^2}}$$

σ^2 : Population variance

$$\alpha = 1 - P(z_{\alpha})$$

Here, $P(z)$ is the c.d.f. value of the standard normal distribution.

C. When the alternative hypothesis is $A > A_0$

For t defined as follows

$$t = \frac{a - A_0}{s_a}$$

$$\begin{cases} \text{If } t \geq z_\alpha, \text{ reject} \\ \text{If } t < z_\alpha, \text{ accept} \end{cases}$$

where s_a is as follows.

$$s_a = \sqrt{\frac{\sigma^2}{\sum (x_i - \mu_x)^2}}$$

σ^2 : Population variance

$$\alpha = 1 - P(z_\alpha)$$

Here, $P(z)$ is the c.d.f. value of the standard normal distribution.

ii. When the population variance is unknown

A. When the alternative hypothesis is $A < A_0$

For t defined as follows

$$t = \frac{a - A_0}{s_a}$$

$$\begin{cases} \text{If } |t| \geq t_{\frac{\alpha}{2}}, \text{ reject} \\ \text{If } |t| < t_{\frac{\alpha}{2}}, \text{ accept} \end{cases}$$

where s_a is as follows.

$$s_a = \sqrt{\frac{\sigma^2}{\sum (x_i - \mu_x)^2}}$$

σ^2 : Unbiased variance of error variation

$$\frac{\alpha}{2} = 1 - P(t_{\frac{\alpha}{2}} | n - 2)$$

Here, $P(t|n)$ is the cumulative distribution function (c.d.f.) of a t distribution having number of degrees of freedom n.

B. When the alternative hypothesis is $A < A_0$

For t defined as follows

$$t = \frac{a - A_0}{s_a}$$

$$\begin{cases} \text{If } t \geq -t_\alpha, \text{ reject} \\ \text{If } t < -t_\alpha, \text{ accept} \end{cases}$$

where s_a is as follows.

$$s_a = \sqrt{\frac{\sigma^2}{\sum (x_i - \mu_x)^2}}$$

σ^2 : Unbiased variance of error variation

$$\alpha = 1 - P(t_\alpha | n - 2)$$

Here, $P(t|n)$ is the cumulative distribution function (c.d.f.) of a t distribution having number of degrees of freedom n.

C. When the alternative hypothesis is $A > A_0$

For t defined as follows

$$t = \frac{a - A_0}{s_a}$$

$$\begin{cases} \text{If } t \geq t_\alpha, \text{ reject} \\ \text{If } t < t_\alpha, \text{ accept} \end{cases}$$

where s_a is as follows.

$$s_a = \sqrt{\frac{\sigma^2}{\sum(x_i - \mu_x)^2}}$$

σ^2 : Unbiased variance of error variation

$$\alpha = 1 - P(t_\alpha | n-2)$$

Here, $P(t|n)$ is the cumulative distribution function (c.d.f.) of a t distribution having number of degrees of freedom n.

(b) Constant term

Hypothesis: $B = B_0$

i. When the population variance is known

A. When the alternative hypothesis is $B \neq B_0$

For t defined as follows

$$t = \frac{b - B_0}{s_b}$$

$$\begin{cases} \text{If } |t| \geq z_{\frac{\alpha}{2}}, \text{ reject} \\ \text{If } |t| < z_{\frac{\alpha}{2}}, \text{ accept} \end{cases}$$

where s_b is as follows.

$$s_b = \sqrt{\sigma^2 \left[\frac{1}{n} + \frac{\mu_x^2}{\sum(x_i - \mu_x)^2} \right]}$$

σ^2 : Population variance

$$\frac{\alpha}{2} = 1 - P(z_{\frac{\alpha}{2}})$$

Here, $P(z)$ is the c.d.f. value of the standard normal distribution.

B. When the alternative hypothesis is $B < B_0$

For t defined as follows

$$t = \frac{b - B_0}{s_b}$$

$$\begin{cases} \text{If } t \geq -z_\alpha, \text{ reject} \\ \text{If } t < -z_\alpha, \text{ accept} \end{cases}$$

where s_b is as follows.

$$s_b = \sqrt{\sigma^2 \left[\frac{1}{n} + \frac{\mu_x^2}{\sum(x_i - \mu_x)^2} \right]}$$

σ^2 : Population variance

$$\alpha = 1 - P(z_\alpha)$$

Here, $P(z)$ is the c.d.f. value of the standard normal distribution.

C. When the alternative hypothesis is $B > B_0$

For t defined as follows

$$t = \frac{b - B_0}{s_b}$$

$$\begin{cases} \text{If } t \geq z_{\frac{\alpha}{2}}, \text{ reject} \\ \text{If } t < z_{\frac{\alpha}{2}}, \text{ accept} \end{cases}$$

where s_b is as follows.

$$s_b = \sqrt{\sigma^2 \left[\frac{1}{n} + \frac{\mu_x^2}{\sum(x_i - \mu_x)^2} \right]}$$

σ^2 : Population variance

$$\alpha = 1 - P(z_\alpha)$$

Here, $P(z)$ is the c.d.f. value of the standard normal distribution.

ii. When the population variance is unknown

A. When the alternative hypothesis is $B \neq B_0$

For t defined as follows

$$t = \frac{b - B_0}{s_b}$$

$$\begin{cases} \text{If } |t| \geq t_{\frac{\alpha}{2}}, \text{ reject} \\ \text{If } |t| < t_{\frac{\alpha}{2}}, \text{ accept} \end{cases}$$

where s_b is as follows.

$$s_b = \sqrt{\sigma^2 \left[\frac{1}{n} + \frac{\mu_x^2}{\sum (x_i - \mu_x)^2} \right]}$$

σ^2 : Unbiased variance of error variation

$$\frac{\alpha}{2} = 1 - P(t_{\frac{\alpha}{2}} | n-2)$$

Here, $P(t|n)$ is the cumulative distribution function (c.d.f.) of a t distribution having number of degrees of freedom n.

B. When the alternative hypothesis is $B < B_0$

For t defined as follows

$$t = \frac{b - B_0}{s_b}$$

$$\begin{cases} \text{If } t \geq -t_{\alpha}, \text{ reject} \\ \text{If } t < -t_{\alpha}, \text{ accept} \end{cases}$$

where s_b is as follows.

$$s_b = \sqrt{\sigma^2 \left[\frac{1}{n} + \frac{\mu_x^2}{\sum (x_i - \mu_x)^2} \right]}$$

σ^2 : Unbiased variance of error variation

$$\alpha = 1 - P(t_{\alpha} | n-2)$$

Here, $P(t|n)$ is the cumulative distribution function (c.d.f.) of a t distribution having number of degrees of freedom n.

C. When the alternative hypothesis is $B > B_0$

For t defined as follows

$$t = \frac{b - B_0}{s_b}$$

$$\begin{cases} \text{If } t \geq t_{\frac{\alpha}{2}}, \text{ reject} \\ \text{If } t < t_{\frac{\alpha}{2}}, \text{ accept} \end{cases}$$

where s_b is as follows.

$$s_b = \sqrt{\sigma^2 \left[\frac{1}{n} + \frac{\mu_x^2}{\sum (x_i - \mu_x)^2} \right]}$$

σ^2 : Unbiased variance of error variation

$$\alpha = 1 - P(t_{\alpha} | n-2)$$

Here, $P(t|n)$ is the cumulative distribution function (c.d.f.) of a t distribution having number of degrees of freedom n.

(2) **Usage**

Double precision:

ierr = ASL_d3tssr (x, n, y, &yv, x0, cl, &ir, z, stat, isw1, isw2, isw3, w);

Single precision:

ierr = ASL_r3tssr (x, n, y, &yv, x0, cl, &ir, z, stat, isw1, isw2, isw3, w);

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	x	{D∗ / R∗}	n	Input	Values of independent variable X of sample, $x_i(i = 1, n)$
2	n	I	1	Input	Number of sample data n
3	y	{D∗ / R∗}	n	Input	Values of dependent variable Y of sample, $y_i(i = 1, n)$
4	yv	{D∗ / R∗}	1	Input	Variance of the population to which dependent variable Y belongs(if isw2 = 1) (See 10.2.1)
				Output	Unbiased variance of error variation σ^2 (When isw2=2)
5	x0	{D / R}	1	Input	When isw1=1 Tested regression coefficient value A_0 When isw1=2 Tested constant term value B_0
6	cl	{D / R}	1	Input	Confidence level $100(1 - \alpha)(\%)$
7	ir	{D∗ / R∗}	1	Output	Test result When isw1=1, ir=0 : Hypothesis $A = A_0$ is accepted ir=1 : Hypothesis $A = A_0$ is rejected When isw1=2, ir=0 : Hypothesis $B = B_0$ is accepted ir=1 : Hypothesis $B = B_0$ is rejected

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
8	z	$\left\{\begin{matrix}\text{D*}\\\text{R*}\end{matrix}\right\}$	2	Output	When isw2=1 and isw3=1 z[0] : Value of t z[1] : Values of a standard normal distribution $z_{\frac{\alpha}{2}}$ When isw2=1 and isw3=2 z[0] : Value of t z[1] : Values of a standard normal distribution $-z_{\alpha}$ When isw2=1 and isw3=3 z[0] : Value of t z[1] : Values of a standard normal distribution z_{α} When isw2=2 and isw3=1 z[0] : Value of t z[1] : Value of t distribution $t_{\frac{\alpha}{2}}$ When isw2=2 and isw3=2 z[0] : Value of t z[1] : Value of t distribution $-t_{\alpha}$ When isw2=2 and isw3=3 z[0] : Value of t z[1] : Value of t distribution t_{α}
9	stat	$\left\{\begin{matrix}\text{D*}\\\text{R*}\end{matrix}\right\}$	2	Output	stat[0] : The sample regression coefficient stat[1] : The sample constant term
10	isw1	I	1	Input	The switch for selection of the statistic. isw1=1 : Testing the population regression coefficient isw1=2 : Testing the population constant term
11	isw2	I	1	Input	Switch for variance isw2=1 : The variance of the population is entered for yv isw2=2 : The variance (not an unbiased estimate) of the sample data is entered for yv

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
12	isw3	I	1	Input	The alternative hypothesis switch For isw1=1, isw3=1 : When the alternative hypothesis is $A \neq A_0$ isw3=2 : When the alternative hypothesis is $A < A_0$ isw3=3 : When the alternative hypothesis is $A > A_0$ For isw1=2, isw3=1 : When the alternative hypothesis is $B \neq B_0$ isw3=2 : When the alternative hypothesis is $B < B_0$ isw3=3 : When the alternative hypothesis is $B > B_0$
13	w	$\begin{Bmatrix} \text{D}* \\ \text{R}* \end{Bmatrix}$	29	Work	Work area
14	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) isw1 $\in \{1, 2\}$

(b) isw2 $\in \{1, 2\}$

(c) isw3 $\in \{1, 2, 3\}$

(d) n ≥ 3

(e) $0.0 \leq$ cl ≤ 100.0

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
1000	isw3=1 : cl=100.0 isw3=2 or 3 : cl=0.0 or cl=100.0.	isw3=1 : The positive maximum value is set for z[1]. isw3=2 or 3 : The positive maximum value or the negative minimum value is set for z[1].
3000	Restriction (a) was not satisfied.	Processing is aborted.
3010	Restriction (b) was not satisfied.	
3020	Restriction (c) was not satisfied.	
3030	Restriction (d) was not satisfied.	
3040	Restriction (e) was not satisfied.	
4000	There are no difference among independent variable X.	
4100	The Unbiased variance of error variation is 0.0. (See 10.2.1)	

(6) **Notes**

None

(7) **Example**

(a) Problem

From the following one set of sample data of size 9, test the hypothesis $A = 0$ related to the population regression coefficient with 95% confidence level.

x_i	y_i
1	3
2	3
3	5
4	5
5	6
6	7
7	8
8	8
9	9

Assume that the alternative hypothesis is $A \neq 0$ and the population variance value is unknown.

(b) Input data

isw1=1 isw2=2, isw3=1, n=9, array x, array y, x0=0.0 and cl=95.0.

(c) Main program

```
/*      C interface example for ASL_d3tssr */

#include <stdio.h>
#include <stdlib.h>
#include <asl.h>

int main()
{
```

```
int n;
double cl;
double yv;
double x0;
int ir;
double z[2];
double stat[2];
double w[29];
int isw1;
int isw2;
int isw3;
int ierr;
double *x;
double *y;
int i;
FILE *fp;

fp = fopen( "d3tssr.dat", "r" );
if( fp == NULL )
{
    printf( "file open error\n" );
    return -1;
}

printf( "    *** ASL_d3tssr ***\n" );
printf( "\n     ** Input **\n\n" );

fscanf( fp, "%d", &isw1);
fscanf( fp, "%d", &isw2);
fscanf( fp, "%d", &isw3);
fscanf( fp, "%d", &n );
fscanf( fp, "%lf", &x0 );
fscanf( fp, "%lf", &cl );

x = ( double * )malloc((size_t)( sizeof(double) * (n) ));
if( x == NULL )
{
    printf( "no enough memory for array x\n" );
    return -1;
}

y = ( double * )malloc((size_t)( sizeof(double) * (n) ));
if( y == NULL )
{
    printf( "no enough memory for array y\n" );
    return -1;
}


for( i=0; i<n; i++ )
{
        fscanf( fp, "%lf", &x[i] );
}
for( i=0; i<n; i++ )
{
        fscanf( fp, "%lf", &y[i] );
}
fclose( fp );

printf( "\n\t**  ASL_d3tssr  **\n");
printf( "\tisw1    = %6d\n", isw1);
printf( "\tisw2    = %6d\n", isw2);
printf( "\tisw3    = %6d\n", isw3);
printf( "\tn       = %6d\n", n);
printf( "\tx0      = %8.3g\n", x0);
printf( "\tcl      = %8.3g\n", cl);
printf("\n\t sample1  sample2\n");
for( i=0; i<n; i++ )
{
    printf( "\t%8.3g %8.3g\n", x[i], y[i]);
}

ierr = ASL_d3tssr(x, n, y, &yv, x0, cl, &ir, z, stat, isw1, isw2, isw3, w);

printf( "\n     ** Output **\n\n" );
printf( "\tierr = %6d\n\n", ierr );

if ( ir == 0 ){
    printf( "\tHypothesis, rho = %8.3g, is accepted.\n ", x0 );
}else{
    printf( "\tHypothesis, rho = %8.3g, is rejected.\n ", x0 );
}
printf( "\n" );
for( i=0 ; i<2 ; i++ )
{
    printf( "\tz[%2d] = %8.3g\n", i, z[i] );
}
printf( "\n" );
printf( "\tRegression coefficient of sample = %8.3g\n", stat[0] );
```

```
    printf( "\tConstant term of sample          = %8.3g\n", stat[1] );

    free( x );
    free( y );

    return 0;
}
```

(d) Output results

```
*** ASL_d3tssr ***

 ** Input **


**  ASL_d3tssr  **
isw1    =      1
isw2    =      2
isw3    =      1
n       =      9
x0      =        0
cl      =       95

 sample1  sample2
       1        3
       2        3
       3        5
       4        5
       5        6
       6        7
       7        8
       8        8
       9        9

 ** Output **

ierr =      0

Hypothesis, rho =        0, is rejected.

z[ 0] =     14.8
z[ 1] =     2.36

Regression coefficient of sample =    0.783
Constant term of sample          =     2.08
```

Chapter 7

ANALYSIS OF VARIANCE AND DESIGN OF EXPERIMENTS

7.1 INTRODUCTION

Design of experiments is one of the fields of inductive statistics. With design of experiments, an actual measurement value x is considered to be the realized value of the random variable X, and the internal structure of X corresponding to the experiment conditions is considered and analyzed through the actual measurement values. To perform the analysis, the experiment must be designed in advance. However, to analyze the actual measurement values, the within class variation or between class variation and error variation quantities are systematically sought by using tables called analysis of variance tables. This type of analysis method is called **analysis of variance**.
This library provides the following functions for performing design of experiments and analysis of variance.

- One-way Layout
- Two-way Layout
- Multiple-way Layout
- Randomized Block Design
- Greco-Latin Square Method
- Cumulative Method
- Balanced Incomplete Block Design (BIBD)

7.1.1 Explanation

(1) **One-way Layout**

With the one-way layout analysis of variance method, given one-way layout data $\{x_{ij}\}(i = 1, \cdots, n_j; j = 1, \cdots, m)$ consisting of m levels so that the number of repetitions in each level is n_j, the problem is to test whether or not there is a difference among the means of m population distributions based on this data. At this time, the structure equation of the one-way layout observed values is defined as follows.

$$x_{ij} = \mu + \alpha_i + \epsilon_{ij}$$

μ represents the total mean over all observed values, α_i represents the effect in level i, and ϵ_{ij} represents the observation error, and the observed values are assumed to be mutually independent values that obey a normal distribution $N(0, \sigma^2)$. The following kinds of data are calculated in the analysis. Mean of each level:

$$\bar{x}_j = \frac{\sum_{i=1}^{n_j} x_{ij}}{n_j} \qquad j = 1, \cdots, m$$

Variance of each level :

$$V_j = \frac{\sum_{i=1}^{n_j} (x_{ij} - \bar{x}_j)^2}{\alpha_j} \qquad j = 1, \cdots, m$$

Total mean :

$$\bar{x} = \frac{\sum_{j=1}^{m} \sum_{i=1}^{n_j} x_{ij}}{\sum_{j=1}^{m} n_j}$$

Here, α_j is n_j when a sample variance is used, and α_j is $n_j - 1$ when an unbiased variance is used.
Variation :

- Total variation

$$S_T = \sum_{j=1}^{m} \sum_{i=1}^{n_j} (x_{ij} - \bar{x})^2$$

- Between class variation

$$S_A = \sum_{j=1}^{m} (\bar{x}_j - \bar{x})^2$$

- Error variation

$$S_E = S_T - S_A$$

Degrees of freedom :

- Degrees of freedom of total variation

$$\phi_T = \sum_{j=1}^{m} n_j - 1$$

- Degrees of freedom of between class variation

$$\phi_A = m - 1$$

- Degrees of freedom of error variation

$$\phi_E = \sum_{j=1}^{m} (n_j - 1)$$

Unbiased variance :

- Unbiased variance of between class variation

$$V_A = \frac{S_A}{\phi_A}$$

- Unbiased variance of error variation

$$V_E = \frac{S_E}{\phi_E}$$

Variance ratio :

$$F_A = \frac{V_A}{V_E}$$

Contribution ratio :

- Contribution ratio of between class variance

$$P_A = \frac{S_A - \phi_A \cdot V_E}{S_T}$$

- Contribution ratio of error variance

$$P_E = 1 - P_A$$

The test is performed by using the critical value of an F distribution for which the variance ratio F_A has degrees of freedom ϕ_A and ϕ_E.

(2) **Two-way Layout**

Given factors A and B consisting of m_a and m_b levels respectively and two-way layout data $\{x_{kij}\}(k = 1, \cdots, n_{ij}; i = 1, \cdots, m_a; j = 1, \cdots, m_b)$ for which the number of repetitions in the combination of each level is n_{ij}, the problem is to test the effect α_i in level i factor A, the effect β_j in level j of factor B, and the interaction effect γ_{ij} when the structure equation of the two-way layout observed values is defined as follows based on this data.

$$x_{kij} = \mu + \alpha_i + \beta_j + \gamma_{ij} + \epsilon_{kij}$$

μ represents the total mean over all observed values, ϵ_{ij} represents the observation error, and the observed values are assumed to be mutually independent values that obey a normal distribution $N(0, \sigma^2)$. Since each effect is a constant, this kind of model is called a **parameter model**. For details, refer to the reference bibliography.

(3) **Multiple-way Layout**

This method is a generalization of the two-way layout method in which the number of factors is increased. The problem is to test the effects of each factor and the interaction effects between the factors. For details, refer to the reference bibliography.

(4) **Randomized Block Design**
Randomized block design is one type of experimental design. Generally, when performing an experiment, there may be factors that tend to increase the efficiency of the experiment if they are taken into account in advance even though they are secondary factors from the viewpoint of conditions (**factors**) having to do with the immediate objective. To eliminate the effect of these kinds of factors, **randomized block design** uses blocks that group the factors so that the effect of these secondary factors is reduced. Generally, the blocks are arranged so that the secondary conditions increase between different blocks and decrease within the same block. With randomized block design, equal subdivisions are provided in the levels of factors in each block, and for each block, the various levels of factors are randomly assigned to each subdivision.

(5) **Greco-Latin Square Method**
The n Latin characters $A, B, \cdots$ arranged in n rows and n columns so that the same Latin character is not duplicated in the same row or same column is called an $n \times n$ Latin Square. By making experimental arrangements using this kind of Latin square to eliminate nonuniformity in the row and column directions, you can test differences in significance among treatments $A, B, \cdots$ (Latin Square Method). However, in this case, interactions between row effects and column effects must not exist, and the general mean, row effect, column effect, treatment effect, and experimental error must be additively combined for the experimental data. An analysis of variance performed using a table obtained by combining two orthogonal Latin Squares (Greco-Latin square) in place of a Latin square is called the Greco-Latin Square Method. The Greco-Latin Square Method can be used when the number of factors is 4, the number of levels for each factor is the same and at least 4, and no interactions of the factors exist.

(6) **Cumulative Method**
For details, refer to the reference bibliography.

(7) **Balanced Incomplete Block design (BIBD)** In a block experiment, when each block contains a complete set of treatments, it is called a complete block design. If this is not the case and one set of treatments to be compared is incomplete and is not entered in the blocks, it is called an incomplete block design. In particular, a block design in which the number of repetitions of each trial is equal and the number of times two arbitrary trials appear in the same block is equal is called a balanced incomplete block design.

7.1.2 Reference Bibliography

(1) Fisher, R. A. , "The design of experiments", 7th ed. , Oliver and Boyd, Edinburgh (1966)

7.2 ONE-WAY LAYOUT

7.2.1 ASL_d41wr1, ASL_r41wr1 One-Way Layout Analysis of Variance

(1) **Function**

Given one-way layout data $\{x_{ij}\}(i = 1, \cdots, n_j; j = 1, \cdots, m)$ consisting of m levels so that the number of repetitions in each level is n_j, the ASL_d41wr1 or ASL_r41wr1 obtains the mean and variance of each level and the total mean over all levels and performs an analysis of variance.

The mean and variance of each level and the total mean over all levels for the one-way layout data $\{x_{ij}\}$ $(i = 1, \cdots, n_j; j = 1, \cdots, m)$ are defined by the following equations.

Mean of each level :

$$\bar{x}_j = \frac{\sum_{i=1}^{n_j} x_{ij}}{n_j} \qquad j = 1, \cdots, m$$

Variance of each level :

$$V_j = \frac{\sum_{i=1}^{n_j} (x_{ij} - \bar{x}_j)^2}{\alpha_j} \qquad j = 1, \cdots, m$$

Total mean :

$$\bar{x} = \frac{\sum_{j=1}^{m} \sum_{i=1}^{n_j} x_{ij}}{\sum_{j=1}^{m} n_j}$$

Here, α_j is n_j when a sample variance is used, and α_j is $n_j - 1$ when an unbiased variance is used.

Also, the analysis of variance results are defined by the following equations.

Variation :

- Total variation

$$S_T = \sum_{j=1}^{m} \sum_{i=1}^{n_j} (x_{ij} - \bar{x})^2$$

- Between class variation

$$S_A = \sum_{j=1}^{m} (\bar{x}_j - \bar{x})^2$$

- Error variation

$$S_E = S_T - S_A$$

Degrees of freedom :

- Degrees of freedom of total variation

$$\phi_T = \sum_{j=1}^{m} n_j - 1$$

- Degrees of freedom of between class variation

$$\phi_A = m - 1$$

- Degrees of freedom of error variation

$$\phi_E = \sum_{j=1}^{m} (n_j - 1)$$

Unbiased variance :

- Unbiased variance of between class variation

$$V_A = \frac{S_A}{\phi_A}$$

- Unbiased variance of error variation

$$V_E = \frac{S_E}{\phi_E}$$

Variance ratio :

$$F_A = \frac{V_A}{V_E}$$

Contribution ratio :

- Contribution ratio of between class variation

$$P_A = \frac{S_A - \phi_A \cdot V_E}{S_T}$$

- Contribution ratio of error variation

$$P_E = 1 - P_A$$

(2) **Usage**

Double precision:

```
ierr = ASL_d41wr1  (a, na, m, n, nr, stat, &x1, v, isw);
```

Single precision:

```
ierr = ASL_r41wr1  (a, na, m, n, nr, stat, &x1, v, isw);
```

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	a	{D∗ / R∗}	na×m	Input	Matrix in which observed values are stored (x_{ij}) (See Note (a))
2	na	I	1	Input	Adjustable dimension of array a
3	m	I	1	Input	Number of levels m
4	n	I*	m	Input	Number of repetitions in jth level n_j (not used when nr$\geq$ 1)
5	nr	I	1	Input	Number of repetitions when the numbers of repetitions in each level are equal $n_1 = n_2 = \cdots = n_g$. When the numbers of repetitions in each level are not equal, set a value that is less than or equal to zero.
6	stat	{D∗ / R∗}	m×2	Output	Mean and variance of each level (See Note (b))
7	x1	{D∗ / R∗}	1	Output	Total mean
8	v	{D∗ / R∗}	11	Output	Analysis of variance results (For the storage method of the analysis of variance results, see Table 7−1 in Note (c).)
9	isw	I	1	Input	Processing switch 0: Calculate the unbiased variance 1: Calculate the sample variance
10	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) isw = 0, 1

(b) nr≤0 and na ≥ n[i] $(\mathrm{i} = 0, \cdots, \mathrm{m}-1)$
or na≥nr

(c) m ≥ 1

(d) nr≤0 and n[i]≥ 1 $(\mathrm{i} = 0, \cdots, \mathrm{m}-1)$
or nr≥1

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
1000	Restriction (a) was not satisfied.	Processing continues with isw=0.
1010	The following five conditions were satisfied at the same time. (a) isw= 0 (b) m> 1 (c) nr< 1 (d) n[i] = 1 for some i (i = 0, ··· , m − 1). (e) n[0]=n[1]= ··· =n[m−1]= 1 does not hold.	The absolute value maximum that can be represented is set for each stat[i−1+m] for which n[i]= 1, and the other stat[i−1+m] are calculated normally.
1020	One of the following conditions was satisfied. (a) m=1 and nr< 1 and n[0]> 1 (b) m= 1 and isw= 1 (c) m> 1 and nr= 1 and isw= 1 (d) m> 1 and nr< 1 and isw= 1 and n[0]=n[1]= ··· =n[m−1]= 1.	The absolute value maximum that can be represented is set for v[6] to v[10].
1030	One of the following conditions was satisfied. (a) nr= 1 and isw= 0 (b) nr< 1 and isw= 0 and n[0]=n[1]= ··· =n[m−1]= 1	The absolute value maximum that can be represented is set for all stat[i − 1 + m] (i= 0, ··· ,m−1) and for v[6] to v[10].
3000	Any of restriction (b), (c) or (d) were not satisfied.	Processing is aborted.

(6) **Notes**

(a) The observed values (x_{ij}) are stored in array a as a real matrix (two-dimensional array type) data (See Appendix A).

(b) The mean and variance of each level are stored as follows in the array stat

stat[$j-1$] : Mean $\bar{x}_j$
stat[$j-1$+m] : Variance v_j , $j = 1, \cdots, m$

(c) The analysis of variance results are stored as follows in the array v.

Table 7−1 Analysis of Variance Table

Element	Variation	Degrees of freedom	Unbiased variance	Variance ratio	Contribution ratio
Total	v[0]	v[3]			
Between level	v[1]	v[4]	v[6]	v[8]	v[9]
Error	v[2]	v[5]	v[7]		v[10]

(d) Statistics obtained when calculations are performed using an unbiased estimate can be applied to a population for which sampling with replacement is performed from an infinite or finite population. Statistics obtained when calculations are performed using a sample variance can be applied to a population for which the population and sample match.

(7) **Example**

(a) Problem

For one-way layout data given by the matrix X shown below, obtain the mean and variance of each level and the total mean over all levels and perform an analysis of variance.

$$X = \begin{bmatrix} 71 & 83 & 85 & 84 & 82 \\ 74 & 79 & 84 & 80 & 78 \\ 76 & 83 & 89 & 82 & 83 \\ 72 & 77 & 82 & 85 & 83 \end{bmatrix}$$

(b) Input data

One-way layout data X, na=100, m=5, nr=4 and isw=0.

(c) Main program

```
/*      C interface example for ASL_d41wr1 */

#include <stdio.h>
#include <stdlib.h>
#include <asl.h>

int main()
{
    double *a;
    int na;
    int m;
    int *n;
    int nr;
    double *stat;
    double x1;
    double v[11];
    int isw;
    int ierr;
    int i,j;
    FILE *fp;

    fp = fopen( "d41wr1.dat", "r" );
    if( fp == NULL )
    {
        printf( "file open error\n" );
        return -1;
    }

    printf( "    *** ASL_d41wr1 ***\n" );
    printf( "\n     ** Input **\n\n" );

    fscanf( fp, "%d", &na );
    fscanf( fp, "%d", &m );
    fscanf( fp, "%d", &nr );
    fscanf( fp, "%d", &isw );

    a = ( double * )malloc((size_t)( sizeof(double) * (na*m) ));
    if( a == NULL )
    {
        printf( "no enough memory for array a\n" );
        return -1;
    }

    stat = ( double * )malloc((size_t)( sizeof(double) * (m*2) ));
    if( stat == NULL )
    {
        printf( "no enough memory for array stat\n" );
        return -1;
    }

    /* Array n is not used when nr is larger than 0 */
    n = ( int * )malloc((size_t)( sizeof(int) * m ));
    if( n == NULL )
    {
        printf( "no enough memory for array n\n" );
        return -1;
    }

    printf( "\tna  = %6d\n", na  );
    printf( "\tm   = %6d\n", m   );
    printf( "\tnr  = %6d\n", nr  );
    printf( "\tisw = %6d\n", isw );

    printf( "\n\tObservations\n\n");
    for( i=0 ; i<nr ; i++ )
    {
        printf( "\t" );
```

```
        for( j=0 ; j<m ; j++ )
        {
            fscanf( fp, "%lf", &a[i+na*j] );
            printf( "%8.3g ", a[i+na*j] );
        }
        printf( "\n" );
    }

    fclose( fp );

    ierr = ASL_d41wr1(a, na, m, n, nr, stat, &x1, v, isw);

    printf( "\n     ** Output **\n\n" );
    printf( "\tierr = %6d\n", ierr );
    printf( "\n\tMean over all levels = %8.3g\n",x1);
    printf("\n\tMean and variance in each level\n\n");
    printf( "\t Level     Mean   Variance\n");
    printf( "\t-------------------------\n");
    for( i=0 ; i<m ; i++ )
    {
        printf("\t%6d %8.3g %8.3g\n",i+1,stat[i],stat[i+m]);
    }
    printf( "\n\tAnalysis of variance table\n\n" );
    printf( "\t Factor      S.S.     D.F.     M.S.    V.R.    C.R.\n" );
    printf( "\t-------------------------------------------------------\n" );
    printf( "\t Total  %8.3g %8.3g\n", v[0],v[3] );
    printf( "\t Level  %8.3g %8.3g %8.3g %8.3g %8.3g\n",
         v[1],v[4],v[6],v[8],v[9]);
    printf( "\t Error  %8.3g %8.3g %8.3g          %8.3g\n",
        v[2],v[5],v[7],v[10]);

    free( a );
    free( stat );

    return 0;
}
```

(d) Output results

```
*** ASL_d41wr1 ***

 ** Input **

na  =    100
m   =      5
nr  =      4
isw =      0

Observations

      71       83       85       84       82
      74       79       84       80       78
      76       83       89       82       83
      72       77       82       85       83

 ** Output **

ierr =      0

Mean over all levels =     80.6

Mean and variance in each level

 Level     Mean   Variance
-------------------------
     1     73.3     4.92
     2     80.5        9
     3       85     8.67
     4     82.8     4.92
     5     81.5     5.67

Analysis of variance table

 Factor      S.S.     D.F.     M.S.    V.R.    C.R.
-------------------------------------------------------
 Total        415       19
 Level        315        4     78.8     11.9    0.696
 Error       99.5       15     6.63              0.304
```

7.3 TWO-WAY LAYOUT

7.3.1 ASL_d42wrn, ASL_r42wrn Two-Way Layout Analysis of Variance

(1) **Function**

For factors A and B consisting of m_a and m_b levels respectively and two-way layout data $\{x_{ij}\}(i = 1, \cdots, m_a; j = 1, \cdots, m_b)$ for which there are no repetitions in the combination of each level, the ASL_d42wrn or ASL_r42wrn obtains the mean and variance of each level for each factor and the total mean over all levels and performs an analysis of variance.

The mean and variance of each level for each factor and the total mean over all levels are defined by the following equations.

Mean of each level of factor A :

$$\bar{x}_{i\cdot} = \frac{1}{m_b} \sum_{j=1}^{m_b} x_{ij}$$

Variance of each level of factor A :

$$V_{ai} = \frac{1}{\alpha_b} \sum_{j=1}^{m_b} (x_{ij} - \bar{x}_{i\cdot})^2$$

Mean of each level of factor B :

$$\bar{x}_{\cdot j} = \frac{1}{m_a} \sum_{i=1}^{m_a} x_{ij}$$

Variance of each level of factor B :

$$V_{bj} = \frac{1}{\alpha_a} \sum_{i=1}^{m_a} (x_{ij} - \bar{x}_{\cdot j})^2$$

Total mean :

$$\bar{x} = \frac{1}{m_a \cdot m_b} \sum_{i=1}^{m_a} \sum_{j=1}^{m_b} x_{ij}$$

Here, $\alpha_a = m_a$ and $\alpha_b = m_b$ when a sample variance is used, and $\alpha_a = m_a - 1$ and $\alpha_b = m_b - 1$ when an unbiased variance is used.

Also, the analysis of variance results are defined by the following equations.

Variation :

- Total variation

$$S_T = \sum_{i=1}^{m_a} \sum_{j=1}^{m_b} (x_{ij} - \bar{x})^2$$

- Variation of factor A

$$S_A = m_b \sum_{i=1}^{m_a} (\bar{x}_{i\cdot} - \bar{x})^2$$

- Variation of factor B

$$S_B = m_a \sum_{j=1}^{m_b} (\bar{x}_{\cdot j} - \bar{x})^2$$

- Error variation

$$S_E = S_T - (S_A + S_B)$$

Degrees of freedom :

- Degrees of freedom of total variation

$$\phi_T = m_a \cdot m_b - 1$$

- Degrees of freedom of factor A variation

$$\phi_A = m_a - 1$$

- Degrees of freedom of factor B variation

$$\phi_B = m_b - 1$$

- Degrees of freedom of error variation

$$\phi_E = (m_a - 1) \cdot (m_b - 1)$$

Unbiased variance :

- Unbiased variance of factor A variation

$$V_A = \frac{S_A}{\phi_A}$$

- Unbiased variance of factor B variation

$$V_B = \frac{S_B}{\phi_B}$$

- Unbiased variance of error variation

$$V_E = \frac{S_E}{\phi_E}$$

Variance ratio :

- Variance ratio for unbiased variance of factor A variation

$$F_A = \frac{V_A}{V_E}$$

- Variance ratio for unbiased variance of factor B variation

$$F_B = \frac{V_B}{V_E}$$

Contribution ratio :

- Contribution ratio of factor A variation

$$P_A = \frac{S_A - \phi_A \cdot V_E}{S_T}$$

- Contribution ratio of factor B variation

$$P_B = \frac{S_B - \phi_B \cdot V_E}{S_T}$$

- Contribution ratio of error variation

$$P_E = 1 - P_A - P_B$$

(2) **Usage**

Double precision:

ierr = ASL_d42wrn (a, na, la, lb, stata, statb, &x1, v, isw);

Single precision:

ierr = ASL_r42wrn (a, na, la, lb, stata, statb, &x1, v, isw);

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	a	{D∗ / R∗}	na×lb	Input	Matrix in which observed values are stored (x_{ij}) (See Note (a))
2	na	I	1	Input	Adjustable dimension of array a
3	la	I	1	Input	Number of levels of factor A m_a
4	lb	I	1	Input	Number of levels of factor B m_b
5	stata	{D∗ / R∗}	la×2	Output	Mean and variance of each level of factor A (See Note (b))
6	statb	{D∗ / R∗}	lb×2	Output	Mean and variance of each level of factor B (See Note (b))
7	x1	{D∗ / R∗}	1	Output	Total mean
8	v	{D∗ / R∗}	16	Output	Analysis of variance table (For the storage method of the analysis of variance table, see Table 7−2 in Note (c).)
9	isw	I	1	Input	Processing switch 0: Calculate the unbiased variance 1: Calculate the sample variance
10	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) isw = 0, 1

(b) na ≥ la ≥ 1

(c) nb ≥ 1

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
1000	Restriction (a) was not satisfied.	Processing continues with isw=0.
1010	isw = 0 was specified when la = 1 or lb = 1.	If la = 1, the absolute value maximum that can be represented is set for the variance of each level of factor B. If lb = 1, the absolute value maximum that can be represented is set for the variance of each level of factor A. Also, the absolute value maximum that can be represented is set for v[8] to v[15].
1020	isw = 1 was specified when la = 1 or lb = 1.	The absolute value maximum that can be represented is set for v[8] to v[15].
3000	Any of restriction (b) or (c) were not satisfied.	Processing is aborted.

(6) **Notes**

(a) The observed values (x_{ij}) are stored in array a as a real matrix (two-dimensional array type) data (See Appendix A).

(b) The mean and variance of each level of factors A and B are stored as follows in the arrays stata and statb.

stata[$i-1$] : Mean of each level of factor A $\bar{x}_{i\cdot}$
stata[$i-1$+la] : Variance of each level of factor A V_{ai}
statb[$j-1$] : Mean of each level of factor B $\bar{x}_{\cdot j}$
statb[$j-1$+lb] : Variance of each level of factor B V_{bj}

, $i = 1, \cdots, \mathrm{la}; j = 1, \cdots, \mathrm{lb}$

(c) The analysis of variance table elements are stored as follows in the array v.

Table 7−2 Analysis of Variance Table Storage Status

Element	Variation	Degrees of freedom	Unbiased variance	Variance ratio	Contribution ratio
Total	v[0]	v[4]			
Factor A	v[1]	v[5]	v[8]	v[11]	v[13]
Factor B	v[2]	v[6]	v[9]	v[12]	v[14]
Error	v[3]	v[7]	v[10]		v[15]

(d) Statistics obtained when calculations are performed using an unbiased estimate can be applied to a population for which sampling with replacement is performed from an infinite or finite population. Statistics obtained when calculations are performed using a sample variance can be applied to a population for which the population and sample match.

(7) **Example**

(a) Problem

Given two-way layout data for which there are no repetitions and having factors A and B as shown in

matrix X below, obtain the mean and variance of each level of each factor and the total mean over all levels and perform an analysis of variance.

$$X = \begin{bmatrix} 1.26 & 1.21 & 1.19 \\ 1.29 & 1.23 & 1.23 \\ 1.38 & 1.27 & 1.22 \end{bmatrix}$$

(b) Input data

Two-way layout data X, na=10, la=3, lb=3 and isw=0.

(c) Main program

```
/*      C interface example for ASL_d42wrn */

#include <stdio.h>
#include <stdlib.h>
#include <asl.h>

int main()
{
    double *a;
    int na, la, lb;
    double *stata, *statb;
    double x1;
    double v[16];
    int isw;
    int ierr;
    int i,j;
    FILE *fp;

    fp = fopen( "d42wrn.dat", "r" );
    if( fp == NULL )
    {
        printf( "file open error\n" );
        return -1;
    }

    printf( "    *** ASL_d42wrn ***\n" );
    printf( "\n     ** Input **\n\n" );

    fscanf( fp, "%d", &na );
    fscanf( fp, "%d", &la );
    fscanf( fp, "%d", &lb );
    fscanf( fp, "%d", &isw );

    a = ( double * )malloc((size_t)( sizeof(double) * (na*lb) ));
    if( a == NULL )
    {
        printf( "no enough memory for array a\n" );
        return -1;
    }

    stata = ( double * )malloc((size_t)( sizeof(double) * (la*2) ));
    if( stata == NULL )
    {
        printf( "no enough memory for array stata\n" );
        return -1;
    }

    statb = ( double * )malloc((size_t)( sizeof(double) * (lb*2) ));
    if( statb == NULL )
    {
        printf( "no enough memory for array statb\n" );
        return -1;
    }

    printf( "\tna  = %6d\n", na  );
    printf( "\tla  = %6d\n", la  );
    printf( "\tlb  = %6d\n", lb  );
    printf( "\tisw = %6d\n", isw );

    printf( "\n\tObservations\n\n" );
    for( i=0 ; i<la ; i++ )
    {
        printf( "\t" );
        for( j=0 ; j<lb ; j++ )
        {
            fscanf( fp, "%lf", &a[i+na*j] );
            printf( "%8.3g ", a[i+na*j] );
        }
        printf( "\n" );
    }

    fclose( fp );
```

```
    ierr = ASL_d42wrn(a, na, la, lb, stata, statb, &x1, v, isw);

    printf( "\n     ** Output **\n\n" );
    printf( "\tierr = %6d\n", ierr );

    printf( "\n\tMean over all levels = %8.3g\n",x1);
    printf( "\n\tMean and variance in each level of factor A\n\n");
    printf( "\t Level     Mean   Variance\n");
    printf( "\t--------------------------\n");
    for( i=0 ; i<lb ; i++ )
    {
        printf("\t%6d %8.3g %8.3g\n",i+1,stata[i],stata[i+la]);
    }
    printf( "\n\tMean and variance in each level of factor B\n\n");
    printf( "\t Level     Mean   Variance\n");
    printf( "\t--------------------------\n");
    for( i=0 ; i<lb ; i++ )
    {
        printf("\t%6d %8.3g %8.3g\n",i+1,statb[i],statb[i+lb]);
    }
    printf( "\n\tAnalysis of variance table\n\n" );
    printf( "\t Factor      S.S.     D.F.     M.S.    V.R.    C.R.\n" );
    printf( "\t-----------------------------------------------------\n" );
    printf( "\t Total  %8.3g %8.3g\n", v[0],v[4] );
    printf( "\t   A    %8.3g %8.3g %8.3g %8.3g %8.3g\n",
        v[1],v[5],v[8],v[11],v[13]);
    printf( "\t   B    %8.3g %8.3g %8.3g %8.3g %8.3g\n",
        v[2],v[6],v[9],v[12],v[14]);
    printf( "\t Error  %8.3g %8.3g %8.3g          %8.3g\n",
        v[3],v[7],v[10],v[15]);

    free( a );
    free( stata );
    free( statb );

    return 0;
}
```

(d) Output results

```
*** ASL_d42wrn ***

 ** Input **

na  =     10
la  =      3
lb  =      3
isw =      0

Observations

    1.26     1.21     1.19
    1.29     1.23     1.23
    1.38     1.27     1.22

 ** Output **

ierr =      0

Mean over all levels =     1.25

Mean and variance in each level of factor A

 Level     Mean   Variance
--------------------------
     1     1.22   0.0013
     2     1.25   0.0012
     3     1.29   0.0067

Mean and variance in each level of factor B

 Level     Mean   Variance
--------------------------
     1     1.31   0.0039
     2     1.24 0.000933
     3     1.21 0.000433

Analysis of variance table

 Factor      S.S.     D.F.      M.S.    V.R.    C.R.
-----------------------------------------------------
 Total    0.0258        8
   A      0.0074        2   0.0037     4.72    0.226
   B      0.0153        2  0.00763     9.74    0.531
 Error   0.00313        4 0.000783             0.243
```

7.3.2 ASL_d42wrm, ASL_r42wrm
Two-Way Layout Analysis of Variance (With Missing Values)

(1) **Function**

For factors A and B consisting of m_a and m_b levels respectively and two-way layout data $\{x_{ij}\}(i = 1, \cdots, m_a; j = 1, \cdots, m_b)$ for which there are no repetitions in the combinations of each level and no data has been obtained for combinations of n_s levels, the ASL_d42wrm or ASL_r42wrm obtains the mean and variance of each level for each factor and the total mean over all levels and performs an analysis of variance. The data values of the combinations of levels for which no data has been obtained are called missing values, and estimates are substituted for the missing values in the calculations of each statistic.

Mean of each level of factor A:

$$\bar{x}_{i\cdot} = \frac{1}{m_b}\sum_{j=1}^{m_b} x_{ij}$$

Variance of each level of factor A:

$$V_{ai} = \frac{1}{\alpha_b}\sum_{j=1}^{m_b}(x_{ij} - \bar{x}_{i\cdot})^2$$

Mean of each level of factor B:

$$\bar{x}_{\cdot j} = \frac{1}{m_a}\sum_{i=1}^{m_a} x_{ij}$$

Variance of each level of factor B:

$$V_{bj} = \frac{1}{\alpha_a}\sum_{i=1}^{m_a}(x_{ij} - \bar{x}_{\cdot j})^2$$

Total mean:

$$\bar{x} = \frac{1}{m_a \cdot m_b}\sum_{i=1}^{m_a}\sum_{j=1}^{m_b} x_{ij}$$

Here, ($\alpha_a = m_a$ and $\alpha_b = m_b$ when a sample variance is used, and $\alpha_a = m_a - 1$ and $\alpha_b = m_b - 1$ when an unbiased variance is used.

Also, the analysis of variance results are defined by the following equations.

Variation:

- Total variation
$$S_T = \sum_{i=1}^{m_a}\sum_{j=1}^{m_b}(x_{ij} - \bar{x})^2$$
- Variation of factor A
$$S_A = m_b\sum_{i=1}^{m_a}(\bar{x}_{i\cdot} - \bar{x})^2$$
- Variation of factor B
$$S_B = m_a\sum_{j=1}^{m_b}(\bar{x}_{\cdot j} - \bar{x})^2$$

- Error variation

$$S_E = S_T - (S_A + S_B)$$

Degrees of freedom:

- Degrees of freedom of total variation

$$\phi_T = m_a \cdot m_b - n_s - 1$$

- Degrees of freedom of factor A variation

$$\phi_A = m_a - 1$$

- Degrees of freedom of factor B variation

$$\phi_B = m_b - 1$$

- Degrees of freedom of error variation

$$\phi_E = (m_a - 1) \cdot (m_b - 1) - n_s$$

Unbiased variance:

- Unbiased variance of factor A variation

$$V_A = \frac{S_A}{\phi_A}$$

- Unbiased variance of factor B variation

$$V_B = \frac{S_B}{\phi_B}$$

- Unbiased variance of error variation

$$V_E = \frac{S_E}{\phi_E}$$

Variance ratio:

- Variance ratio for unbiased variance of factor A variation

$$F_A = \frac{V_A}{V_E}$$

- Variance ratio for unbiased variance of factor B variation

$$F_B = \frac{V_B}{V_E}$$

Contribution ratio:

- Contribution ratio of factor A variation

$$P_A = \frac{S_A - \phi_A \cdot V_E}{S_T}$$

- Contribution ratio of factor B variation

$$P_B = \frac{S_B - \phi_B \cdot V_E}{S_T}$$

- Contribution ratio of factor error variation

$$P_E = 1 - P_A - P_B$$

Missing value estimates:
The missing value estimates are determined so that the error variation S_E is minimized. To obtain estimates that will minimize S_E, you should solve the following equation with the missing values x_{st} $((s,t) \in S)$ as unknowns.

$$\frac{\partial S_E}{\partial x_{st}} = 0 \qquad ((s,t) \in S)$$

Here, S is assumed to be the set of combinations of levels for which missing values occurred. Since S_E is a quadratic equation, this equation is a set of n_s simultaneous linear equations with the missing values as unknowns.
For example, for the following two-way layout data in which x_{13} and x_{22} are missing values,

$$X = \begin{bmatrix} 1.0 & 1.1 & x_{13} \\ 1.2 & x_{22} & 1.3 \\ 1.1 & 1.0 & 1.3 \end{bmatrix}$$

the error variation is as follows.

$$S_E = 1.78 - 1.4x_{13} - 1.4x_{22} + \frac{4}{9}x_{13}^2 + \frac{4}{9}x_{22}^2 + \frac{2}{9}x_{13}x_{22}$$

Differentiating this with respect to x_{13} and x_{22} produces the following simultaneous linear equations.

$$\frac{\partial S_E}{\partial x_{13}} = -1.4 + \frac{8}{9}x_{13} + \frac{2}{9}x_{22} = 0$$

$$\frac{\partial S_E}{\partial x_{22}} = -1.4 + \frac{2}{9}x_{13} + \frac{8}{9}x_{22} = 0$$

Solving these simultaneous linear equations yields the missing value estimates $x_{13} = 1.26$ and $x_{22} = 1.26$.

(2) **Usage**

Double precision:

```
ierr = ASL_d42wrm  (a, na, la, lb, ist, isn, stata, statb, &x1, v, isw, iwk, wk);
```

Single precision:

```
ierr = ASL_r42wrm  (a, na, la, lb, ist, isn, stata, statb, &x1, v, isw, iwk, wk);
```

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	a	{D∗ / R∗}	na×lb	Input	Matrix in which observed values are stored (x_{ij}) (See Note (b)).
				Output	Matrix in which observed values are stored (x_{ij}). However, estimates are stored in the elements corresponding to missing values.
2	na	I	1	Input	Adjustable dimension of array a.
3	la	I	1	Input	Number of levels of factor A m_a
4	lb	I	1	Input	Number of levels of factor B m_b
5	ist	I*	isn×2	Input	Information about combinations of levels for which missing values occurred. (See Note (a))
6	isn	I	1	Input	Number of missing values n_s
7	stata	{D∗ / R∗}	la×2	Output	Mean and variance of each level of factor A (See Note (c))
8	statb	{D∗ / R∗}	lb×2	Output	Mean and variance of each level of factor B (See Note (c))
9	x1	{D∗ / R∗}	1	Output	Total mean
10	v	{D∗ / R∗}	16	Output	Analysis of variance table (For the storage method of the analysis of variance table, see Table 7−3.)
11	isw	I	1	Input	Processing switch 0: Calculate the unbiased variance 1: Calculate the sample variance
12	iwk	I*	See Contents	Work	Work area **Size:**la × lb + isn
13	wk	{D∗ / R∗}	See Contents	Work	Work area **Size:**$\text{isn}^2 + 2 \times \text{isn} + \text{la} + \text{lb} + 1$
14	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) isw = 0, 1

(b) na $\geq$ la ≥ 2

(c) lb ≥ 2

(d) $1 \leq$ isn $<$ (la -1) $\times$ (lb -1)

(e) $1 \leq$ ist[i -1] $\leq$ la (i $= 1, 2, \cdots,$ isn)

(f) $1 \leq$ ist[i $-1+$ isn] $\leq$ lb (i $= 1, 2, \cdots,$ isn)

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
1000	Restriction (a) was not satisfied.	Processing continues with isw=0.
2000	Estimates of missing values could not be calculated.	The missing value data is replaced by the mean of the non-missing value data, and processing continues.
3000	Any of restrictions (b) to (f) was not satisfied.	Processing is aborted.

(6) **Notes**

(a) The factor A level of the i-th missing value is stored in ist[i -1] and the factor B level is stored in ist[isn + i -1].

(b) The observed values (x_{ij}) are stored in array a as a real matrix (two-dimensional array type) (See Appendix A).

(c) The mean and variance of each level of factors A and B are stored as follows in the arrays stata and statb.

stata[$i-1$] : Mean of each level of factor A $\bar{x}_{i\cdot}$
stata[$i-1+$la] : Variance of each level of factor A V_{ai}
statb[$j-1$] : Mean of each level of factor B $\bar{x}_{\cdot j}$
statb[$j-1+$lb] : Variance of each level of factor B V_{bj}

, $i = 1, \cdots,$ la; $j = 1, \cdots,$ lb

(d) The analysis of variance table elements are stored as follows in the array v.

Table 7−3 Analysis of Variance Table Storage Status

Element	Variation	Degrees of freedom	Unbiased variance	Variance ratio	Contribution ratio
Total	v[0]	v[4]			
Factor A	v[1]	v[5]	v[8]	v[11]	v[13]
Factor B	v[2]	v[6]	v[9]	v[12]	v[14]
Error	v[3]	v[7]	v[10]		v[15]

(e) Statistics obtained when an unbiased variance is calculated can be applied to a population for which sampling with replacement is performed from an infinite or finite population. Statistics obtained when a sample variance is calculated can be applied to a population for which the population and sample match.

(7) **Example**

(a) Problem

Given two-way layout data for which there are no repetitions and having factors A and B as shown in matrix X below, obtain the mean and variance of each level of each factor and the total mean over all levels and perform an analysis of variance. However, $*$ indicates that the corresponding data is a missing value.

$$X = \begin{bmatrix} 4 & 6 & 8 & 10 & 12 \\ 7 & 10 & * & 16 & * \\ * & 14 & 18 & 22 & 26 \\ 13 & 18 & 23 & 28 & 33 \end{bmatrix}$$

(b) Input data

Two-way layout data X, na=4, la=4, lb=5, isn=3, isw=0,
ist[0]=2, ist[isn]=3, ist[1]=4, ist[isn+1]=1 and ist[1]=2, ist[isn+1]=5.

(c) Main program

```
/*      C interface example for ASL_d42wrm */

#include <stdio.h>
#include <stdlib.h>
#include <asl.h>

int main()
{
    double *a;
    int na, la, lb;
    int isn,*ist;
    double *stata, *statb;
    double x1;
    double v[16];
    double *wk;
    int isw;
    int *iwk;
    int ierr;
    int i,j;
    FILE *fp;

    fp = fopen( "d42wrm.dat", "r" );
    if( fp == NULL )
    {
        printf( "file open error\n" );
        return -1;
    }

    printf( "    *** ASL_d42wrm ***\n" );
    printf( "\n     ** Input **\n\n" );

    fscanf( fp, "%d", &isw );
    fscanf( fp, "%d", &na );
    fscanf( fp, "%d", &la );
    fscanf( fp, "%d", &lb );
    fscanf( fp, "%d", &isn );

    a = ( double * )malloc((size_t)( sizeof(double) * (na*lb) ));
    if( a == NULL )
    {
        printf( "no enough memory for array a\n" );
        return -1;
    }

    stata = ( double * )malloc((size_t)( sizeof(double) * (la*2) ));
    if( stata == NULL )
    {
        printf( "no enough memory for array stata\n" );
        return -1;
    }

    statb = ( double * )malloc((size_t)( sizeof(double) * (lb*2) ));
    if( statb == NULL )
    {
        printf( "no enough memory for array statb\n" );
        return -1;
    }

    ist = ( int * )malloc((size_t)( sizeof(int) * (isn*2) ));
    if( ist == NULL )
    {
```

```
    printf( "no enough memory for array ist\n" );
    return -1;
}

iwk = ( int * )malloc((size_t)( sizeof(int) * (la*lb+isn) ));
if( iwk == NULL )
{
    printf( "no enough memory for array iwk\n" );
    return -1;
}

wk = ( double * )malloc((size_t)(sizeof(double)*(isn*isn+2*isn+la+lb+1)));
if( statb == NULL )
{
    printf( "no enough memory for array wk\n" );
    return -1;
}

printf( "\tisw = %6d\n", isw );
printf( "\tna  = %6d\n", na  );
printf( "\tla  = %6d\n", la  );
printf( "\tlb  = %6d\n", lb  );
printf( "\tisn = %6d\n", isn );

printf( "\n\tMissed values\n\n" );
for( i=0 ; i<isn ; i++ )
{

    fscanf( fp, "%d %d", &ist[i], &ist[i+isn] );
    printf( "\t  a(%6d,%6d)\n", ist[i], ist[i+isn] );
}
printf( "\n\tObservations\n\n" );
for( i=0 ; i<la ; i++ )
{
    printf( "\t" );
    for( j=0 ; j<lb ; j++ )
    {
        fscanf( fp, "%lf", &a[i+na*j] );
        printf( "%8.3g ", a[i+na*j] );
    }
    printf( "\n" );
}

fclose( fp );

ierr = ASL_d42wrm(a,na,la,lb,ist,isn,stata,statb,&x1,v,isw,iwk,wk);

printf( "\n     ** Output **\n\n" );
printf( "\tierr = %6d\n", ierr );

printf( "\n\tEstimated missed values\n\n" );
for( i=0 ; i<isn ; i++ )
{

    printf( "\t  a(%6d,%6d) = %8.3g\n",
        ist[i], ist[i+isn], a[ist[i]-1+na*(ist[i+isn]-1)] );
}
printf( "\n\tMean over all levels = %8.3g\n",x1);
printf( "\n\tMean and variance in each level of factor A\n\n");
printf( "\t Level     Mean   Variance\n");
printf( "\t---------------------------\n");
for( i=0 ; i<la ; i++ )
{
    printf("\t%6d %8.3g %8.3g\n",i+1,stata[i],stata[i+la]);
}
printf( "\n\tMean and variance in each level of factor B\n\n");
printf( "\t Level     Mean   Variance\n");
printf( "\t---------------------------\n");
for( i=0 ; i<lb ; i++ )
{
    printf("\t%6d %8.3g %8.3g\n",i+1,statb[i],statb[i+lb]);
}
printf( "\n\tAnalysis of variance table\n\n" );
printf( "\t Factor      S.S.     D.F.     M.S.    V.R.    C.R.\n" );
printf( "\t-----------------------------------------------------\n" );
printf( "\t Total  %8.3g %8.3g\n", v[0],v[4] );
printf( "\t   A    %8.3g %8.3g %8.3g %8.3g %8.3g\n",
    v[1],v[5],v[8],v[11],v[13]);
printf( "\t   B    %8.3g %8.3g %8.3g %8.3g %8.3g\n",
    v[2],v[6],v[9],v[12],v[14]);
printf( "\t Error  %8.3g %8.3g %8.3g          %8.3g\n",
    v[3],v[7],v[10],v[15]);

free( a );
free( stata );
free( statb );
free( ist );
free( iwk );
```

```
    free( wk );

    return 0;
}
```

(d) Output results

```
*** ASL_d42wrm ***

 ** Input **

isw =      0
na  =      4
la  =      4
lb  =      5
isn =      3

Missed values

  a(     2,     3)
  a(     4,     1)
  a(     2,     5)

Observations

       4        6        8       10       12
       7       10        0       16        0
       0       14       18       22       26
      13       18       23       28       33

 ** Output **

ierr =      0

Estimated missed values

  a(     2,     3) =     14.4
  a(     4,     1) =     14.3
  a(     2,     5) =     21.7

Mean over all levels =     15.3

Mean and variance in each level of factor A

 Level     Mean   Variance
-------------------------
     1        8         10
     2     13.8       32.2
     3       16        100
     4     23.3       56.2

Mean and variance in each level of factor B

 Level     Mean   Variance
-------------------------
     1     6.33       36.6
     2       12       26.7
     3     15.9       39.8
     4       19         60
     5     23.2       77.1

Analysis of variance table

 Factor      S.S.     D.F.     M.S.    V.R.    C.R.
-----------------------------------------------------
 Total  1.39e+03       16
   A         597        3      199    14.5    0.399
   B         670        4      168    12.2    0.442
 Error       124        9     13.8            0.158
```

7.3.3 ASL_d42wr1, ASL_r42wr1 Two-Way Layout Analysis of Variance (Repetition Data)

(1) **Function**

For factors A and B consisting of m_a and m_b levels respectively and two-way layout data $\{x_{kij}\}(k = 1, \cdots, n_{ij}; i = 1, \cdots, m_a; j = 1, \cdots, m_b)$ for which the number of repetitions in the combination of each level is n_{ij}, the ASL_d42wr1 or ASL_r42wr1 obtains the mean and variance of each level for each factor and the total mean over all levels and performs an analysis of variance.

The mean of the repetitions in the combination of each level, the mean and variance of each level for each factor, and the total mean over all levels are defined by the following equations.

Mean of the repetitions in the combination of each level :

$$\bar{x}_{\cdot ij} = \frac{1}{n_{ij}} \sum_{k=1}^{n_{ij}} x_{kij}$$

Mean of each level of factor A :

$$\bar{x}_{\cdot i \cdot} = \frac{1}{\sum_{j=1}^{m_b} n_{ij}} \sum_{j=1}^{m_b} \sum_{k=1}^{n_{ij}} x_{kij}$$

Variance of each level of factor A :

$$V_{ai} = \frac{1}{\alpha_i} \sum_{j=1}^{m_b} \sum_{k=1}^{n_{ij}} (x_{kij} - \bar{x}_{\cdot i \cdot})^2$$

Mean of each level of factor B :

$$\bar{x}_{\cdot \cdot j} = \frac{1}{\sum_{i=1}^{m_a} n_{ij}} \sum_{i=1}^{m_a} \sum_{k=1}^{n_{ij}} x_{kij}$$

Variance of each level of factor B :

$$V_{bj} = \frac{1}{\beta_j} \sum_{i=1}^{m_a} \sum_{k=1}^{n_{ij}} (x_{kij} - \bar{x}_{\cdot \cdot j})^2$$

Total mean :

$$\bar{x} = \frac{1}{\sum_{i=1}^{m_a} \sum_{j=1}^{m_b} n_{ij}} \sum_{i=1}^{m_a} \sum_{j=1}^{m_b} \sum_{k=1}^{n_{ij}} x_{kij}$$

Here, $alpha_i$ and $beta_j$ are given by

$$\alpha_i = \sum_{j=1}^{m_b} n_{ij}, \quad \beta_j = \sum_{i=1}^{m_a} n_{ij}$$

when a sample variance is used, and $alpha_i$ and $beta_j$ are given by

$$\alpha_i = \sum_{j=1}^{m_b} n_{ij} - 1, \quad \beta_j = \sum_{i=1}^{m_a} n_{ij} - 1$$

when an unbiased variance is used.
Also, the analysis of variance results are defined by the following equations.
Variation :

- Total variation
$$S_T = \sum_{i=1}^{m_a}\sum_{j=1}^{m_b}\sum_{k=1}^{n_{ij}}(x_{kij} - \bar{x})^2$$
- Variation of factor A
$$S_A = \sum_{i=1}^{m_a}\sum_{j=1}^{m_b} n_{ij}(\bar{x}_{\cdot i\cdot} - \bar{x})^2$$
- Variation of factor B
$$S_B = \sum_{i=1}^{m_a}\sum_{j=1}^{m_b} n_{ij}(\bar{x}_{\cdot\cdot j} - \bar{x})^2$$
- Interaction variation
$$S_{A\times B} = S_{AB} - (S_A + S_B)$$
Here,
$$S_{AB} = \sum_{i=1}^{m_a}\sum_{j=1}^{m_b} n_{ij}(\bar{x}_{\cdot ij} - \bar{x})^2$$
- Error variation
$$S_E = S_T - S_{AB}$$

Degrees of freedom :

- Degrees of freedom of total variation
$$\phi_T = m_a \cdot m_b - 1$$
- Degrees of freedom of factor A variation
$$\phi_A = m_a - 1$$
- Degrees of freedom of factor B variation
$$\phi_B = m_b - 1$$
- Degrees of freedom of interaction variation
$$\phi_{A\times B} = (m_a - 1)(m_b - 1)$$
- Degrees of freedom of error variation
$$\phi_E = \sum_{i=1}^{m_a}\sum_{j=1}^{m_b} n_{ij} - m_a \cdot m_b$$

Unbiased variance :

- Unbiased variance of factor A variation
$$V_A = \frac{S_A}{\phi_A}$$

- Unbiased variance of factor B variation

$$V_B = \frac{S_B}{\phi_B}$$

- Unbiased variance of interaction variation

$$V_{A\times B} = \frac{S_{A\times B}}{\phi_{A\times B}}$$

- Unbiased variance of error variation

$$V_E = \frac{S_E}{\phi_E}$$

Variance ratio :

- Variance ratio for unbiased variance of factor A variation

$$F_A = \frac{V_A}{V_E}$$

- Variance ratio for unbiased variance of factor B variation

$$F_B = \frac{V_B}{V_E}$$

- Variance ratio for unbiased variance of interaction variation

$$F_{A\times B} = \frac{V_{A\times B}}{V_E}$$

Contribution ratio :

- Contribution ratio of factor A variation

$$P_A = \frac{S_A - \phi_A \cdot V_E}{S_T}$$

- Contribution ratio of factor B variation

$$P_B = \frac{S_B - \phi_B \cdot V_E}{S_T}$$

- Contribution ratio of interaction variation

$$P_{A\times B} = \frac{S_{A\times B} - \phi_{A\times B} \cdot V_E}{S_T}$$

- Contribution ratio of error variation

$$P_E = 1 - P_A - P_B - P_{A\times B}$$

(2) **Usage**

Double precision:

```
ierr = ASL_d42wr1  (a, na, ma, la, lb, n, nr, y, stata, statb, &x1, v, isw, wk);
```

Single precision:

```
ierr = ASL_r42wr1  (a, na, ma, la, lb, n, nr, y, stata, statb, &x1, v, isw, wk);
```

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	a	{D∗ / R∗}	See Contents	Input	Observed values (x_{kij}) (See Note (a)) **Size**: na×ma×lb
2	na	I	1	Input	Adjustable dimension of first dimension of array a
3	ma	I	1	Input	Adjustable dimension of second dimension of array a
4	la	I	1	Input	Number of levels of factor A m_a
5	lb	I	1	Input	Number of levels of factor B m_b
6	n	I*	ma×lb	Input	Number of repetitions in the combination (i, j) of each level n_{ij} (not used when nr ≥ 1)
7	nr	I	1	Input	Number of repetitions when the numbers of repetitions in the combination of each level are equal $n_{11} = \cdots = n_{1m_b} = n_{21} = \cdots = n_{2m_b} = \cdots = n_{m_a1} = \cdots = n_{m_am_b}$ When the numbers of repetitions in the combination of each level are not equal, set a value that is less than or equal to zero.
8	y	{D∗ / R∗}	ma×lb	Output	Mean of the repetitions in the combination (i, j) of each level $\bar{x}_{\cdot ij}$
9	stata	{D∗ / R∗}	la×2	Output	Mean and variance of each level of factor A (See Note (b))
10	statb	{D∗ / R∗}	lb×2	Output	Mean and variance of each level of factor B (See Note (b))
11	x1	{D∗ / R∗}	1	Output	Total mean $\bar{x}$
12	v	{D∗ / R∗}	21	Output	Analysis of variance table (For the storage method of the analysis of variance table, see Table 7−4 in Note (c).)

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
13	isw	I	1	Input	Processing switch 0: Calculate the unbiased variance 1: Calculate the sample variance
14	wk	$\left\{\begin{matrix}\text{D}*\\ \text{R}*\end{matrix}\right\}$	la	Work	Work area
15	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) isw = 0, 1

(b) ma $\geq$ la $\geq$ 1

(c) lb $\geq$ 1

(d) nr<1 and na$\geq$n[$i + j\times$la] $\geq$ 1 $\quad (i = 0, \cdots$,la-1 ; $j =$0, . . . ,lb$-1)$
or na$\geq$nr$\geq$ 1

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
1000	Restriction (a) was not satisfied.	Processing continues with isw=0.
1010	la=1 or lb=1 was specified.	The absolute value maximum that can be represented is set for v[10] to v[20]. For the values stored in stata and statb. (See Note (e))
3000	Any of restriction (b) to (d) was not satisfied.	Processing is aborted.

(6) **Notes**

(a) Observed values x_{kij} are stored as follows in the array a.
a[k-1+na$\times$ (i-1+ma$\times$(k-1))]=x_{kij} $(k = 1, \cdots, n_{ij}; i = 1, \cdots, m_a; j = 1, \cdots, m_b)$

(b) The mean and variance of each level of factors A and B are stored as follows in the arrays stata and statb.

stata[$i-1$] : Mean of each level of factor A $\bar{x}_{.i.}$
stata[$i-1$+la] : Variance of each level of factor A V_{ai}
statb[$j-1$] : Mean of each level of factor B $\bar{x}_{..j}$
statb[$j-1$+lb] : Variance of each level of factor B V_{bj}
, $\quad i = 1, \cdots, \text{la}; j = 1, \cdots, \text{lb}$

(c) The analysis of variance table elements are stored as follows in the array v.

Table 7−4 Analysis of Variance Table Storage Status

Factor	Variation	Degrees of freedom	Unbiased variance	Variance ratio	Contribution ratio
Total	v[0]	v[5]			
Factor A	v[1]	v[6]	v[10]	v[14]	v[17]
Factor B	v[2]	v[7]	v[11]	v[15]	v[18]
A×B	v[3]	v[8]	v[12]	v[16]	v[19]
Error	v[4]	v[9]	v[13]		v[20]

(d) Statistics obtained when calculations are performed using an unbiased estimate can be applied to a population for which sampling with replacement is performed from an infinite or finite population. Statistics obtained when calculations are performed using a sample variance can be applied to a population for which the population and sample match.

(e) If ierr=1010 occurred when isw=0 was specified, the following processing is performed.

i. When nr=1 and la=1 :
The absolute value maximum that can be represented is set for all statb[$j-1$+lb] ($j=1,\cdots$,lb).

ii. When nr=1 and lb=1 :
The absolute value maximum that can be represented is set for all stata[$i-1$+la] ($i=1,\cdots$,la).

iii. When n[$j-1$]=1 for a j for which nr<1 and la=1:
The absolute value maximum that can be represented is set for statb[$j-1$+la].

iv. When n[$i-1$]=1 for an i for which nr<1 and lb=1:
The absolute value maximum that can be represented is set for stata[$i-1$+la].

v. When none of the conditions in 6(e)i. to 6(e)iv. occurs:
All stata[$i-1$+la] ($i=1,\cdots$,la) and statb[$j-1$+lb]($j=1,\cdots$,lb) are calculated normally.

(7) **Example**

(a) Problem
Given two-way layout data for which the number of repetitions is 2 and having factors A and B as shown in matrices X_1 and X_2 below, obtain the mean in the combinations of each level, mean and variance of each level of each factor, and the total mean over all levels and perform an analysis of variance.

$$X_1 = \begin{bmatrix} 7.9 & 9.8 & 13.2 & 13.1 \\ 10.3 & 14.6 & 15.9 & 9.5 \\ 9.1 & 12.1 & 11.1 & 7.0 \end{bmatrix}$$

$$X_2 = \begin{bmatrix} 8.7 & 9.3 & 14.0 & 12.0 \\ 11.0 & 14.0 & 14.6 & 8.5 \\ 8.6 & 12.9 & 10.0 & 8.2 \end{bmatrix}$$

(b) Input data
Two-way layout data X_1 and X_2,
na=100, ma=5, la=3, lb=4, nr=2 and isw=0.

(c) Main program

```
/*      C interface example for ASL_d42wr1 */

#include <stdio.h>
#include <stdlib.h>
#include <asl.h>

int main()
{
    double *a;
    int na, ma, la, lb;
    int *n;
    int nr;
    double *y;
    double *stata, *statb;
    double x1;
    double v[21];
    int isw;
    double *wk;
    int ierr;
    int i,j;
    FILE *fp;

    fp = fopen( "d42wr1.dat", "r" );
    if( fp == NULL )
    {
        printf( "file open error\n" );
        return -1;
    }

    printf( "    *** ASL_d42wr1 ***\n" );
    printf( "\n     ** Input **\n\n" );

    fscanf( fp, "%d", &na );
    fscanf( fp, "%d", &ma );
    fscanf( fp, "%d", &la );
    fscanf( fp, "%d", &lb );
    fscanf( fp, "%d", &nr );
    fscanf( fp, "%d", &isw );

    a = ( double * )malloc((size_t)( sizeof(double) * (na*ma*lb) ));
    if( a == NULL )
    {
        printf( "no enough memory for array a\n" );
        return -1;
    }

    y = ( double * )malloc((size_t)( sizeof(double) * (ma*lb) ));
    if( y == NULL )
    {
        printf( "no enough memory for array y\n" );
        return -1;
    }

    stata = ( double * )malloc((size_t)( sizeof(double) * (la*2) ));
    if( stata == NULL )
    {
        printf( "no enough memory for array stata\n" );
        return -1;
    }

    statb = ( double * )malloc((size_t)( sizeof(double) * (lb*2) ));
    if( statb == NULL )
    {
        printf( "no enough memory for array statb\n" );
        return -1;
    }

    wk = ( double * )malloc((size_t)( sizeof(double) * la ));
    if( wk == NULL )
    {
        printf( "no enough memory for array wk\n" );
        return -1;
    }

    /* Array n is not used when nr is larger than 0 */
    n = ( int * )malloc((size_t)( sizeof(int) * (ma*lb) ));
    if( n == NULL )
    {
        printf( "no enough memory for array n\n" );
        return -1;
    }

    printf( "\tna  = %6d\n", na  );
    printf( "\tma  = %6d\n", ma  );
    printf( "\tla  = %6d\n", la  );
    printf( "\tlb  = %6d\n", lb  );
    printf( "\tnr  = %6d\n", nr  );
    printf( "\tisw = %6d\n", isw );
```

```
    printf("\n\tObservations\n");
    printf("\n\t1st time in repetition\n\n");
    for( i=0 ; i<la ; i++ )
    {
        printf( "\t" );
        for( j=0 ; j<lb ; j++ )
        {
            fscanf( fp, "%lf", &a[na*i+na*ma*j] );
            printf( "%8.3g", a[na*i+na*ma*j] );
        }
        printf( "\n" );
    }
    printf("\n\t2nd time in repetition\n\n");
    for( i=0 ; i<la ; i++ )
    {
        printf( "\t" );
        for( j=0 ; j<lb ; j++ )
        {
            fscanf( fp, "%lf", &a[1+na*i+na*ma*j] );
            printf( "%8.3g", a[1+na*i+na*ma*j] );
        }
        printf( "\n" );
    }

    fclose( fp );

    ierr = ASL_d42wr1(a, na, ma, la, lb, n, nr, y,
             stata, statb, &x1, v, isw, wk);

    printf( "\n     ** Output **\n\n" );
    printf( "\tierr = %6d\n", ierr );

    printf( "\n\tMean over all levels = %8.3g\n",x1);
    printf( "\n\tMean for repetition\n\n" );
    for( i=0 ; i<la ; i++ )
    {
        printf("\t");
        for( j=0 ; j<lb ; j++ )
        {
            printf( "%8.3g ", y[i+ma*j]);
        }
        printf("\n");
    }
    printf("\n\tMean and variance in each level of factor A\n\n");
    printf( "\t Level     Mean   Variance\n");
    printf( "\t--------------------------\n");
    for( i=0 ; i<la ; i++ )
    {
        printf("\t%6d %8.3g %8.3g\n",i+1,stata[i],stata[i+la]);
    }
    printf("\n\tMean and variance in each level of factor B\n\n");
    printf( "\t Level     Mean   Variance\n");
    printf( "\t--------------------------\n");
    for( i=0 ; i<lb ; i++ )
    {
        printf("\t%6d %8.3g %8.3g\n",i+1,statb[i],statb[i+lb]);
    }
    printf( "\n\tAnalysis of variance table\n\n" );
    printf( "\t Factor      S.S.     D.F.     M.S.    V.R.    C.R.\n" );
    printf( "\t-----------------------------------------------------\n" );
    printf( "\t Total  %8.3g %8.3g\n", v[0],v[5] );
    printf( "\t   A    %8.3g %8.3g %8.3g %8.3g %8.3g\n",
        v[1],v[6],v[10],v[14],v[17]);
    printf( "\t   B    %8.3g %8.3g %8.3g %8.3g %8.3g\n",
        v[2],v[7],v[11],v[15],v[18]);
    printf( "\t A X B  %8.3g %8.3g %8.3g %8.3g %8.3g\n",
        v[3],v[8],v[12],v[16],v[19]);
    printf( "\t Error  %8.3g %8.3g %8.3g          %8.3g\n",
        v[4],v[9],v[13],v[20]);

    free( a );
    free( y );
    free( stata );
    free( statb );
    free( wk );
    free( n );

    return 0;
}
```

(d) Output results

```
*** ASL_d42wr1 ***

 ** Input **

na  =    100
ma  =      5
la  =      3
lb  =      4
nr  =      2
isw =      0

Observations

1st time in repetition

     7.9     9.8    13.2    13.1
    10.3    14.6    15.9     9.5
     9.1    12.1    11.1       7

2nd time in repetition

     8.7     9.3      14      12
      11      14    14.6     8.5
     8.6    12.9      10     8.2

 ** Output **

ierr =      0

Mean over all levels =     11.1

Mean for repetition

     8.3     9.55     13.6     12.6
    10.7     14.3     15.3        9
    8.85     12.5     10.6      7.6

Mean and variance in each level of factor A

 Level     Mean   Variance
--------------------------
     1       11       5.5
     2     12.3     7.77
     3     9.88     4.13

Mean and variance in each level of factor B

 Level     Mean   Variance
--------------------------
     1     9.27     1.35
     2     12.1     4.73
     3     13.1      4.9
     4     9.72     5.57

Analysis of variance table

 Factor      S.S.     D.F.     M.S.    V.R.    C.R.
-------------------------------------------------------
 Total        145       23
   A         23.6        2     11.8    28.8    0.156
   B         62.6        3     20.9      51    0.422
 A X B       54.3        6     9.04    22.1    0.356
 Error       4.91       12    0.409          0.0647
```

7.4 MULTIPLE-WAY LAYOUT

7.4.1 ASL_d4mwrf, ASL_r4mwrf Multiple-Way Layout Analysis of Variance

(1) **Function**

For m factors $A_1, A_2, \cdots, A_m$ (where m is at most 6) consisting of $l_1, l_2, \cdots, l_m$ levels respectively and multi-way layout data $x_{kj_1j_2\cdots j_m}(k = 1, \cdots, n; i = 1, \cdots, m; j_i = 1, \cdots, l_i)$ for which the number of repetitions in the combination of each factor level is a fixed value n, the ASL_d4mwrf or ASL_r4mwrf performs an analysis of variance. At this time, specified interactions can be confounded. For the explanations below, the operations Σ_i and Δ_i are defined as follows.

$$\Sigma_i \equiv \sum_{j_i=1}^{l_i}, \quad \Delta_i \equiv l_i - \sum_{j_i=1}^{l_i}$$

The mean of the repetitions in the combination of each level and the total mean over all levels are defined by the following equations.

Mean of the repetitions in the combination of each level :

$$\bar{x}_{\cdot j_1 j_2 \cdots j_m} = \frac{1}{n} \sum_{k=1}^{n} x_{kj_1j_2\cdots j_m}$$

Total mean :

$$\bar{x} = \frac{1}{n \prod_{i=1}^{m} l_i} \Sigma_1 \Sigma_2 \cdots \Sigma_m \sum_{k=1}^{n} x_{kj_1j_2\cdots j_m}$$

Also, the analysis of variance results are defined by the following equations.

Variation :

- Total variation

$$S_T = \Sigma_1 \Sigma_2 \cdots \Sigma_m \sum_{k=1}^{n} (x_{kj_1j_2\cdots j_m} - \bar{x})^2$$

- Variation of factor A_i

$$S_{A_i} = \frac{n}{l_i \prod_{j=1}^{m} l_j} \Sigma_i (\Sigma_1 \Sigma_2 \cdots \Sigma_{i-1} \Delta_i \Sigma_{i+1} \cdots \Sigma_m \bar{x}_{\cdot j_1 j_2 \cdots j_m})^2$$

- Variation of the interaction $A_{i_1} \times A_{i_2} \times \cdots \times A_{i_s}$ is of $s(s \leq m)$ factors $A_{i_1}, A_{i_2}, \cdots, A_{i_s}$

$$\begin{aligned} S_{A_{i_1} \times A_{i_2} \times \cdots \times A_{i_s}} &= \frac{n}{(\prod_{k=1}^{s} l_{i_s}) \cdot (\prod_{k=1}^{m} l_k)} \\ &\times \Sigma_{i_1} \Sigma_{i_2} \cdots \Sigma_{i_s} (\Sigma_1 \cdots \Sigma_{i_1 - 1} \Delta_{i_1} \Sigma_{i_1 + 1} \cdots \\ &\quad \Sigma_{i_2 - 1} \Delta_{i_2} \Sigma_{i_2 + 1} \cdots \Sigma_{i_s - 1} \Delta_{i_s} \Sigma_{i_s + 1} \cdots \Sigma_m \bar{x}_{\cdot j_1 j_2 \cdots \mathrm{J} m})^2 \end{aligned}$$

- Error variation
  When there are no repetitions :

$$\begin{aligned} S_E &= S_T - (\text{Sum of variations of each factor}) \\ &- (\text{Sum of variations of interactions other than the highest order interaction } [A_1 \times A_2 \times \cdots \times A_m]) \end{aligned}$$

  When there are repetitions :

$$S_E = S_T - (\text{Sum of variations of each factor}) - (\text{Sum of variations of interactions})$$

Degrees of freedom :

- Degrees of freedom of total variation

$$\phi_T = n \prod_{i=1}^{m} l_i - 1$$

- Degrees of freedom of factor A_i variation

$$\phi_A = l_i - 1$$

- Degrees of freedom of variation of the interaction $A_{i_1} \times A_{i_2} \times \cdots \times A_{i_s}$

$$\phi_{A_{i_1} \times A_{i_2} \times \cdots \times A_{i_s}} = \prod_{k=1}^{s} (l_{i_k} - 1)$$

- Degrees of freedom of error variation
  When there are no repetitions :

$$\begin{aligned} \phi_E &= \phi_T - (\text{Sum of degrees of freedom of variations of each factor}) \\ &- (\text{Sum of degrees of freedom of variations of interactions other than the highest order interaction}) \end{aligned}$$

  When there are repetitions :

$$\begin{aligned} \phi_E &= \phi_T - (\text{Sum of degrees of freedom of variations of each factor}) \\ &- (\text{Sum of degrees of freedom of variations of interactions}) \end{aligned}$$

Unbiased variance :

- Unbiased variance of factor A_i variation

$$V_{A_i} = \frac{S_{A_i}}{\phi_{A_i}}$$

- Unbiased variance of variation of the interaction $A_{i_1} \times A_{i_2} \times \cdots \times A_{i_s}$

$$V_{A_{i_1} \times A_{i_2} \times \cdots \times A_{i_s}} = \frac{S_{A_{i_1} \times A_{i_2} \times \cdots \times A_{i_s}}}{\phi_{A_{i_1} \times A_{i_2} \times \cdots \times A_{i_s}}}$$

  However, the unbiased variance of the variation of the highest order interaction can be defined only when there are repetitions.

- Unbiased variance of error variation

$$V_E = \frac{S_E}{\phi_E}$$

Variance ratio :

- Variance ratio for unbiased variance of factor A_i variation

  $$F_{A_i} = \frac{V_{A_i}}{V_E}$$

- Variance ratio for unbiased variance of variation of the interaction $A_{i_1} \times A_{i_2} \times \cdots \times A_{i_s}$

  $$F_{A_{i_1} \times A_{i_2} \times \cdots \times A_{i_s}} = \frac{V_{A_{i_1} \times A_{i_2} \times \cdots \times A_{i_s}}}{V_E}$$

  However, the variance ratio of the variation of the highest order interaction can be defined only when there are repetitions.

Contribution ratio :

- Contribution ratio of factor A_i variation

  $$P_{A_i} = \frac{S_{A_i} - \phi_{A_i} \cdot V_E}{S_T}$$

- Contribution ratio of variation of the interaction $A_{i_1} \times A_{i_2} \times \cdots \times A_{i_s}$

  $$P_{A_{i_1} \times A_{i_2} \times \cdots \times A_{i_s}} = \frac{S_{A_{i_1} \times A_{i_2} \times \cdots \times A_{i_s}} - \phi_{A_{i_1} \times A_{i_2} \times \cdots \times A_{i_s}} \cdot V_E}{S_T}$$

- Contribution ratio of error variation
  When there are no repetitions :

  $$\begin{aligned} P_E &= \phi_T - (\text{Sum of contribution ratios of variations of each factor}) \\ &- (\text{Sum of contribution ratios of variations of interactions other than the highest order interaction}) \end{aligned}$$

  When there are repetitions :

  $$\begin{aligned} P_E &= \phi_T - (\text{Sum of contribution ratios of variations of each factor}) \\ &- (\text{Sum of contribution ratios of variations of interactions}) \end{aligned}$$

(2) **Usage**

Double precision:

```
ierr = ASL_d4mwrf  (a, na, n, lt, m, ipt, ipn, y, &x1, v, wk);
```

Single precision:

```
ierr = ASL_r4mwrf  (a, na, n, lt, m, ipt, ipn, y, &x1, v, wk);
```

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	a	{D∗ / R∗}	na×m1	Input	Observed values ($x_{k\beta}$) (See Note (a)) Here, m1 = $\prod_{i=1}^{m} lt[i-1]$
2	na	I	1	Input	Adjustable dimension of array a
3	n	I	1	Input	Number of repetitions in the combination of each level n
4	lt	I*	m	Input	Number of levels of each factor l_i
5	m	I	1	Input	Number of factors m
6	ipt	I*	m2	Input	Factor numbers of factors to be confounded (See Note (b)) Here, when ipn>0, m2=ipn, and when ipn=0, the argument ipt should be a dummy argument.
7	ipn	I	1	Input	Number of factors to be confounded
8	y	{D∗ / R∗}	m1	Output	Mean of repetitions in combination of each level $\bar{x}_{.\beta}$. (See Note (a).) Here, m1 = $\prod_{i=1}^{m} lt[i-1]$
9	x1	{D∗ / R∗}	1	Output	Total mean $\bar{x}$
10	v	{D∗ / R∗}	$(2^m+1)\times5$	Output	Analysis of variance table (For the storage method of the analysis of variance table, see Table 7−5 in note (c).)
11	wk	{D∗ / R∗}	m3	Work	Work area. Here, m3 = $\prod_{i=1}^{n}(lt[i-1]+1)$
12	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) $na \geq n \geq \begin{cases} 2 & (m=1) \\ 1 & (m>1) \end{cases}$

(b) $1 \leq m \leq 6$

(c) $lt[i] \geq 1 \quad (i = 0, \cdots, m-1)$

(d) $0 \leq ipn \leq \begin{cases} 2^m - 2 & (n=1) \\ 2^m - 1 & (n>1) \end{cases}$

(e) $2 \leq ipt[i] \leq \begin{cases} 2^m - 1 & (n=1) \\ 2^m & (n>1) \end{cases} \quad (i = 0, \cdots, m-1)$

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
1000	lt[i] was specified for some i ($i = 0, 1, \cdots, \mathrm{m}-1$).	The absolute value maximum that can be represented is set for items other than the variation and degrees of freedom items of the analysis of variance table.
3000	Any of restriction (a) to (e) was not satisfied.	Processing is aborted.

(6) **Notes**

(a) Multi-way layout data having m factors $A_1, A_2, \cdots, A_m$ for which the number of repetitions in the combination of each factor level is a fixed value n is represented using $m+1$ subscripts as follows.

$$x_{kj_1j_2\cdots j_m} \ (k = 1, \cdots, n; i = 1, \cdots, m; j_i = 1, \cdots, l_i)$$

However, in this function, the multi-way layout data is represented using 2 subscripts as follows, and that data is stored in array a as a real matrix (two-dimensional array type) (See Appendix A).

$$x_{kj_1j_2\cdots j_m} \to x_{k\beta}$$

Here, β is defined as follows.

$$\beta = j_1 + \sum_{i=2}^{m}(j_i \prod_{k=1}^{i-1} l_k)$$

Similarly, the mean of the repetitions in the combination of each level, which is given by

$$\bar{x}_{\cdot j_1j_2\cdots j_m} \ (i = 1, \cdots, m; j_i = 1, \cdots, l_i)$$

can be represented by one subscript as follows and stored in array y as a one-dimensional vector.

$$\bar{x}_{\cdot j_1j_2\cdots j_m} \to \bar{x}_{\cdot\beta}$$

Here, β is defined as follows.

$$\beta = j_1 + \sum_{i=2}^{m}(j_i \prod_{k=1}^{i-1} l_k)$$

(b) The factors that are subject to an analysis of variance are numbered as follows.

- One-way layout

Factor number	Factor
1	Total
2	A_1
3	Error

- Two-way layout

Factor number	Factor
1	Total
2	A_1
3	A_2
4	$A_1 \times A_2$
5	Error

- Three-way layout

Factor number	Factor
1	Total
2	A_1
3	A_2
4	$A_1 \times A_2$
5	A_3
6	$A_1 \times A_3$
7	$A_2 \times A_3$
8	$A_1 \times A_2 \times A_3$
9	Error

- Four-way layout

Factor number	Factor
1	Total
2	A_1
3	A_2
4	$A_1 \times A_2$
5	A_3
6	$A_1 \times A_3$
7	$A_2 \times A_3$
8	$A_1 \times A_2 \times A_3$
9	A_4
10	$A_1 \times A_4$
11	$A_2 \times A_4$
12	$A_1 \times A_2 \times A_4$
13	$A_3 \times A_4$
14	$A_1 \times A_3 \times A_4$
15	$A_2 \times A_3 \times A_4$
16	$A_1 \times A_2 \times A_3 \times A_4$
17	Error

- Five-way layout

Factor number	Factor
1	Total
2	A_1
3	A_2
4	$A_1 \times A_2$
5	A_3
6	$A_1 \times A_3$
7	$A_2 \times A_3$
8	$A_1 \times A_2 \times A_3$
9	A_4
10	$A_1 \times A_4$
11	$A_2 \times A_4$
12	$A_1 \times A_2 \times A_4$
13	$A_3 \times A_4$
14	$A_1 \times A_3 \times A_4$
15	$A_2 \times A_3 \times A_4$
16	$A_1 \times A_2 \times A_3 \times A_4$
17	A_5
18	$A_1 \times A_5$
19	$A_2 \times A_5$
20	$A_1 \times A_2 \times A_5$
21	$A_3 \times A_5$
22	$A_1 \times A_3 \times A_5$
23	$A_2 \times A_3 \times A_5$
24	$A_1 \times A_2 \times A_3 \times A_5$
25	$A_4 \times A_5$
26	$A_1 \times A_4 \times A_5$
27	$A_2 \times A_4 \times A_5$
28	$A_1 \times A_2 \times A_4 \times A_5$
29	$A_3 \times A_4 \times A_5$
30	$A_1 \times A_3 \times A_4 \times A_5$
31	$A_2 \times A_3 \times A_4 \times A_5$
32	$A_1 \times A_2 \times A_3 \times A_4 \times A_5$
33	Error

- Six-way layout

Factor number	Factor	Factor number	Factor
1	Total	34	$A_1 \times A_6$
2	A_1	35	$A_2 \times A_6$
3	A_2	36	$A_1 \times A_2 \times A_6$
4	$A_1 \times A_2$	37	$A_3 \times A_6$
5	A_3	38	$A_1 \times A_3 \times A_6$
6	$A_1 \times A_3$	39	$A_2 \times A_3 \times A_6$
7	$A_2 \times A_3$	40	$A_1 \times A_2 \times A_3 \times A_6$
8	$A_1 \times A_2 \times A_3$	41	$A_4 \times A_6$
9	A_4	42	$A_1 \times A_4 \times A_6$
10	$A_1 \times A_4$	43	$A_2 \times A_4 \times A_6$
11	$A_2 \times A_4$	44	$A_1 \times A_2 \times A_4 \times A_6$
12	$A_1 \times A_2 \times A_4$	45	$A_3 \times A_4 \times A_6$
13	$A_3 \times A_4$	46	$A_1 \times A_3 \times A_4 \times A_6$
14	$A_1 \times A_3 \times A_4$	47	$A_2 \times A_3 \times A_4 \times A_6$
15	$A_2 \times A_3 \times A_4$	48	$A_1 \times A_2 \times A_3 \times A_4 \times A_6$
16	$A_1 \times A_2 \times A_3 \times A_4$	49	$A_5 \times A_6$
17	A_5	50	$A_1 \times A_5 \times A_6$
18	$A_1 \times A_5$	51	$A_2 \times A_5 \times A_6$
19	$A_2 \times A_5$	52	$A_1 \times A_2 \times A_5 \times A_6$
20	$A_1 \times A_2 \times A_5$	53	$A_3 \times A_5 \times A_6$
21	$A_3 \times A_5$	54	$A_1 \times A_3 \times A_5 \times A_6$
22	$A_1 \times A_3 \times A_5$	55	$A_2 \times A_3 \times A_5 \times A_6$
23	$A_2 \times A_3 \times A_5$	56	$A_1 \times A_2 \times A_3 \times A_5 \times A_6$
24	$A_1 \times A_2 \times A_3 \times A_5$	57	$A_4 \times A_5 \times A_6$
25	$A_4 \times A_5$	58	$A_1 \times A_4 \times A_5 \times A_6$
26	$A_1 \times A_4 \times A_5$	59	$A_2 \times A_4 \times A_5 \times A_6$
27	$A_2 \times A_4 \times A_5$	60	$A_1 \times A_2 \times A_4 \times A_5 \times A_6$
28	$A_1 \times A_2 \times A_4 \times A_5$	61	$A_3 \times A_4 \times A_5 \times A_6$
29	$A_3 \times A_4 \times A_5$	62	$A_1 \times A_3 \times A_4 \times A_5 \times A_6$
30	$A_1 \times A_3 \times A_4 \times A_5$	63	$A_2 \times A_3 \times A_4 \times A_5 \times A_6$
31	$A_2 \times A_3 \times A_4 \times A_5$	64	$A_1 \times A_2 \times A_3 \times A_4 \times A_5 \times A_6$
32	$A_1 \times A_2 \times A_3 \times A_4 \times A_5$	65	Error
33	A_6		

(c) The analysis of variance table elements are stored as follows in the array v.

Table 7−5 Analysis of Variance Table Storage Status

Factor number	Variation	Degrees of freedom	Unbiased variance	Variance ratio	Contribution ratio
1	v[0]	v[n1 + 1]	*	*	*
2	v[1]	v[n1 + 1]	v[2 × n1 + 1]	v[3 × n1 + 1]	v[4 × n1 + 1]
3	v[2]	v[n1 + 2]	v[2 × n1 + 2]	v[3 × n1 + 2]	v[4 × n1 + 2]
⋮	⋮	⋮	⋮	⋮	⋮
n1 − 1	v[n1 − 2]	v[2 × n1 − 2]	v[3 × n1 − 2]	v[4 × n1 − 2]	v[5 × n1 − 2]
n1	v[n1 − 1]	v[2 × n1 − 1]	v[3 × n1 − 1]	*	v[5 × n1 − 1]

Here, $n1 = 2^m + 1$. The absolute value maximum that can be represented is set for array elements v[2 × n1 + 1], v[3 × n1 + 1], v[4 × n1 + 1], and v[4 × n1 − 1], which are elements for which there is no corresponding analysis of variance table item. Also, when there are no repetitions, 0.0 is set for the value of each item for the highest order interaction factor (factor number n1 − 1). In addition, for the error, 0.0 is set for the value of each item for the factors specified in array ipt, which are to be confounded.

(7) **Example**

(a) Problem

Perform an analysis of variance for three-way layout data for which the number of levels for each factor is 3 and the number of repetitions is 2. Assume that the transpose matrix X^T of the observation matrix X is given as follows.

$$\begin{bmatrix} 5.5 & 5.4 \\ 6.3 & 6.5 \\ 6.9 & 6.8 \\ 5.4 & 5.3 \\ 6.5 & 6.3 \\ 6.9 & 6.5 \\ 5.5 & 5.3 \\ 6.5 & 6.2 \\ 7.0 & 6.8 \\ 4.9 & 4.6 \\ 5.7 & 5.6 \\ 6.2 & 6.0 \\ 5.0 & 5.2 \\ 6.1 & 6.5 \\ 6.6 & 6.5 \\ 4.8 & 4.2 \\ 5.7 & 5.4 \\ 6.9 & 6.4 \\ 4.2 & 4.0 \\ 5.0 & 4.9 \\ 5.4 & 5.3 \\ 4.5 & 4.3 \\ 5.2 & 5.0 \\ 6.1 & 6.0 \\ 4.9 & 4.3 \\ 5.3 & 5.2 \\ 6.4 & 6.3 \end{bmatrix}$$

(b) Input data

Observation matrix X, na=2, n=2, m=3, ipn=0, lt[0]=3, lt[1]=3 and lt[2]=3.

(c) Main program

```
/*      C interface example for ASL_d4mwrf */

#include <stdio.h>
#include <stdlib.h>
#include <asl.h>

int main()
{
    double *a;
    int na;
    int n;
    int *lt;
    int m;
    int *ipt;
    int ipn;
    double *y;
    double x1;
    double *v;
    double *wk;
    int ierr;
    int i,m1,nv,m3;
    FILE *fp;

    fp = fopen( "d4mwrf.dat", "r" );
    if( fp == NULL )
    {
        printf( "file open error\n" );
        return -1;
    }

    printf( "    *** ASL_d4mwrf ***\n" );
```

```
printf( "\n     ** Input **\n\n" );

fscanf( fp, "%d", &na );
fscanf( fp, "%d", &n );
fscanf( fp, "%d", &m );
fscanf( fp, "%d", &ipn );

printf( "\tna=%2d,  n=%2d,  m=%2d,  ipn=%2d\n", na, n, m, ipn );

lt = ( int * )malloc((size_t)( sizeof(int) * m ));
if( lt == NULL )
{
    printf( "no enough memory for array lt\n" );
    return -1;
}

printf( "\n\tNumber of level of each factor\n\n" );
m1 = 1;
m3 = 1;
nv = 1;
printf( "\t" );
for( i=0 ; i<m ; i++)
{
    fscanf( fp, "%d", &lt[i] );
    printf( "%6d ",lt[i] );
    m1 *= lt[i];
    m3 *= lt[i]+1;
    nv *= 2;
}
printf( "\n" );
nv += 1;

a = ( double * )malloc((size_t)( sizeof(double) * (na*m1) ));
if( a == NULL )
{
    printf( "no enough memory for array a\n" );
    return -1;
}

y = ( double * )malloc((size_t)( sizeof(double) * m1 ));
if( y == NULL )
{
    printf( "no enough memory for array y\n" );
    return -1;
}

v = ( double * )malloc((size_t)( sizeof(double) * nv * 5 ));
if( v == NULL )
{
    printf( "no enough memory for array v\n" );
    return -1;
}

wk = ( double * )malloc((size_t)( sizeof(double) * m3 ));
if( wk == NULL )
{
    printf( "no enough memory for array wk\n" );
    return -1;
}

/* Array ipt is not used when ipn is equal to 0 */
ipt=NULL;

printf( "\n\tObservations\n\n" );
printf( "\t 1st time in repetition\n" );
for( i=0 ; i<m1 ; i++ )
{
    if( i%5 == 0 )
    {
        printf( "\n\t" );
    }
    fscanf( fp, "%lf", &a[i*na] );
    printf( "%8.3g", a[i*na] );
}
printf( "\n\n" );

printf( "\t 2nd time in repetition\n" );
for( i=0 ; i<m1 ; i++ )
{
    if( i%5 == 0 )
    {
        printf( "\n\t" );
    }
    fscanf( fp, "%lf", &a[1+i*na] );
    printf( "%8.3g", a[1+i*na] );
}
printf( "\n" );

fclose( fp );

ierr = ASL_d4mwrf(a, na, n, lt, m, ipt, ipn, y, &x1, v, wk);
```

```
    printf( "\n     ** Output **\n\n" );
    printf( "\tierr = %6d\n", ierr );
    printf( "\n\tMean over all levels = %8.3g\n", x1 );

    printf( "\n\tMean for repetition\n" );
    for( i=0 ; i<m1 ; i++ )
    {
        if( i%5 == 0 )
        {
            printf( "\n\t" );
        }
        printf( "%8.3g", y[i] );
    }

    printf( "\n\n\tAnalysis of variance table\n\n" );
    printf( "\t Factor      S.S.     D.F.     M.S.    V.R.    C.R.\n" );
    printf( "\t-------------------------------------------------\n" );
    printf( "\t Total  %8.3g %8.3g\n", v[0],v[nv] );
    for( i=1 ; i<nv-1 ; i++)
    {
        printf( "\t %6d %8.3g %8.3g %8.3g %8.3g %8.3g\n",
         i+1,v[i],v[nv+i],v[2*nv+i],v[3*nv+i],v[4*nv+i]);
    }
    printf( "\t Error  %8.3g %8.3g %8.3g          %8.3g\n",
    v[nv-1],v[2*nv-1],v[3*nv-1],v[5*nv-1]);

    free( a );
    free( lt );
    free( y );
    free( v );
    free( wk );

    return 0;
}
```

(d) Output results

```
*** ASL_d4mwrf ***

 ** Input **

na= 2,  n= 2,  m= 3,  ipn= 0

Number of level of each factor

     3      3      3

Observations

 1st time in repetition

     5.5     6.3     6.9     5.4     6.5
     6.9     5.5     6.5       7     4.9
     5.7     6.2       5     6.1     6.6
     4.8     5.7     6.9     4.2       5
     5.4     4.5     5.2     6.1     4.9
     5.3     6.4

 2nd time in repetition

     5.4     6.5     6.8     5.3     6.3
     6.5     5.3     6.2     6.8     4.6
     5.6       6     5.2     6.5     6.5
     4.2     5.4     6.4       4     4.9
     5.3     4.3       5       6     4.3
     5.2     6.3

 ** Output **

ierr =      0

Mean over all levels =     5.67

Mean for repetition

    5.45     6.4    6.85    5.35     6.4
     6.7     5.4    6.35     6.9    4.75
    5.65     6.1     5.1     6.3    6.55
     4.5    5.55    6.65     4.1    4.95
    5.35     4.4     5.1    6.05     4.6
    5.25    6.35

Analysis of variance table

 Factor      S.S.     D.F.     M.S.    V.R.    C.R.
-------------------------------------------------
 Total       36.1       53
      2      21.6        2     10.8      289    0.596
      3     0.703        2    0.351     9.39   0.0174
      4     0.513        4    0.128     3.43   0.0101
      5      10.4        2     5.18      138    0.285
```

6	0.314	4	0.0785	2.1	0.00456
7	1.25	4	0.313	8.37	0.0305
8	0.356	8	0.0445	1.19	0.00157
Error	1.01	27	0.0374		0.0549

7.4.2 ASL_d4mwrm, ASL_r4mwrm Multiple-Way Layout Analysis of Variance (With Missing Values)

(1) **Function**

For m factors $A_1, A_2, \cdots, A_m$ (where m is at most 6) consisting of $l_1, l_2, \cdots, l_m$ levels respectively and multi-way layout data $x_{j_1 j_2 \cdots j_m} (i = 1, \cdots, m; j_i = 1, \cdots, l_i)$ for which there are no repetitions in the combinations of levels of each factor and no data has been obtained for combinations of n_s combinations of levels, the ASL_d4mwrm or ASL_r4mwrm performs an analysis of variance. At this time, specified interactions can be confounded. The data values of the combinations of levels for which no data has been obtained are called missing values, and estimates are substituted for the missing values in the calculations of each statistic.
For the explanations below, the operations Σ_i and Δ_i for factor A_i are defined as follows.

$$\Sigma_i \equiv \sum_{j_i=1}^{l_i}, \quad \Delta_i \equiv l_i - \sum_{j_i=1}^{l_i}$$

The total mean over all levels is defined by the following equation.
Total mean:

$$\bar{x} = \frac{1}{\prod_{i=1}^{m} l_i} \Sigma_1 \Sigma_2 \cdots \Sigma_m \sum_{k=1}^{n} x_{k j_1 j_2 \cdots j_m}$$

Also, the analysis of variance results are defined by the following equations.
Variation:

- Total variation

$$S_T = \Sigma_1 \Sigma_2 \cdots \Sigma_m \sum_{k=1}^{n} (x_{k j_1 j_2 \cdots j_m} - \bar{x})^2$$

- Variation of factor A_i

$$S_{A_i} = \frac{1}{l_i \prod_{j=1}^{m} l_j} \Sigma_i (\Sigma_1 \Sigma_2 \cdots \Sigma_{i-1} \Delta_i \Sigma_{i+1} \cdots \Sigma_m x_{j_1 j_2 \cdots j_m})^2$$

- Variation of the interaction $A_{i_1} \times A_{i_2} \times \cdots \times A_{i_s}$ of $s (s < m)$ factors $A_{i_1}, A_{i_2}, \cdots, A_{i_s}$.

$$\begin{aligned} S_{A_{i_1} \times A_{i_2} \times \cdots \times A_{i_s}} &= \frac{n}{(\prod_{k=1}^{s} l_{i_s}) \cdot (\prod_{k=1}^{m} l_k)} \\ &\times \Sigma_{i_1} \Sigma_{i_2} \cdots \Sigma_{i_s} (\Sigma_1 \cdots \Sigma_{i_1 - 1} \Delta_{i_1} \Sigma_{i_1 + 1} \cdots \\ &\quad \Sigma_{i_2 - 1} \Delta_{i_2} \Sigma_{i_2 + 1} \cdots \Sigma_{i_s - 1} \Delta_{i_s} \Sigma_{i_s + 1} \cdots \Sigma_m x_{j_1 j_2 \cdots \mathrm{J}_m})^2 \end{aligned}$$

However, the variation of the highest order interaction $A_1 \times A_2 \times \cdots \times A_m$.

- Error variation

$$\begin{aligned} S_E &= S_T - (\text{Sum of variations of each factor}) \\ &- (\text{Sum of variations of interactions other than the highest order interaction} \\ &\quad [A_1 \times A_2 \times \cdots \times A_m]) \end{aligned}$$

Degrees of freedom:

- Degrees of freedom of total variation

$$\phi_T = n \prod_{i=1}^{m} l_i - n_s - 1$$

- Degrees of freedom of factor A_i variation

$$\phi_A = l_i - 1$$

- Degrees of freedom of variation of the interaction $A_{i_1} \times A_{i_2} \times \cdots \times A_{i_s}$

$$\phi_{A_{i_1} \times A_{i_2} \times \cdots \times A_{i_s}} = \prod_{k=1}^{s} (l_{i_k} - 1)$$

- Degrees of freedom of error variation

$$\begin{aligned} \phi_E &= \phi_T - (\text{Sum of degrees of freedom of variations of each factor}) \\ &- (\text{Sum of degrees of freedom of variations of interactions other than the highest order interaction -}n_s) \end{aligned}$$

Unbiased variance:

- Unbiased variance of factor A_i variation

$$V_{A_i} = \frac{S_{A_i}}{\phi_{A_i}}$$

- Unbiased variance of variation of the interaction $A_{i_1} \times A_{i_2} \times \cdots \times A_{i_s}$

$$V_{A_{i_1} \times A_{i_2} \times \cdots \times A_{i_s}} = \frac{S_{A_{i_1} \times A_{i_2} \times \cdots \times A_{i_s}}}{\phi_{A_{i_1} \times A_{i_2} \times \cdots \times A_{i_s}}}$$

  However, the unbiased variance of the variation of the highest order interaction is not defined.

- Unbiased variance of error variation

$$V_E = \frac{S_E}{\phi_E}$$

Variance ratio:

- Variance ratio for unbiased variance of factor A_i variation

$$F_{A_i} = \frac{V_{A_i}}{V_E}$$

- Variance ratio for unbiased variance of variation of the interaction $A_{i_1} \times A_{i_2} \times \cdots \times A_{i_s}$

$$F_{A_{i_1} \times A_{i_2} \times \cdots \times A_{i_s}} = \frac{V_{A_{i_1} \times A_{i_2} \times \cdots \times A_{i_s}}}{V_E}$$

  However, the variance ratio of the variation of the highest order interaction is not defined.

Contribution ratio:

- Contribution ratio of factor A_i variation

$$P_{A_i} = \frac{S_{A_i} - \phi_{A_i} \cdot V_E}{S_T}$$

- Contribution ratio of variation of the interaction $A_{i_1} \times A_{i_2} \times \cdots \times A_{i_s}$

$$P_{A_{i_1} \times A_{i_2} \times \cdots \times A_{i_s}} = \frac{S_{A_{i_1} \times A_{i_2} \times \cdots \times A_{i_s}} - \phi_{A_{i_1} \times A_{i_2} \times \cdots \times A_{i_s}} \cdot V_E}{S_T}$$

  However, the contribution ratio of the variation of the highest order interaction is not defined.

- Contribution ratio of error variation

$$\begin{aligned} P_E &= \phi_T - (\text{Sum of contribution ratios of variations of each factor}) \\ &- (\text{Sum of contribution ratios of variations of interactions other than the highest order interaction}) \end{aligned}$$

Missing value estimates:
The missing value estimates are determined so that the error variation S_E is minimized. To obtain estimates that will minimize S_E, you should solve the following equation with the missing values

$$x_{s_1 s_2 \ldots s_m} \ ((s_1, s_2, \ldots, s_m) \in S)$$

as unknowns.

$$\frac{\partial S_E}{\partial x_{st}} = 0 \qquad ((s,t) \in S)$$

Here, S is assumed to be the set of combinations of levels for which missing values occurred. Since S_E is a quadratic equation, this equation is a set of n_s simultaneous linear equations with the missing values as unknowns.
For example, for the following three-way layout data in which there are three levels of each of the factors A, B, and C and x_{132} and x_{322} are missing values,

$$X_1 = \begin{bmatrix} 11 & 12 & 10 \\ 11 & 10 & 12 \\ 12 & x_{132} & 13 \end{bmatrix} \qquad \text{Factor A level 1 data}$$

$$X_2 = \begin{bmatrix} 11 & 11 & 12 \\ 13 & 14 & 11 \\ 9 & 10 & 12 \end{bmatrix} \qquad \text{Factor A level 2 data}$$

$$X_3 = \begin{bmatrix} 10 & 12 & 11 \\ 10 & x_{322} & 12 \\ 12 & 8 & 11 \end{bmatrix} \qquad \text{Factor A level 3 data}$$

the error variation is as follows.

$$\begin{aligned} S_E &= S_T - S_A - S_B - S_C - S_{A\times B} - S_{B\times C} - S_{C\times A} \\ &= 69714 - 5400x_{132} - 5724x_{322} + 216x_{132}^2 + 216x_{322} + 108x_{132}x_{322} \end{aligned}$$

Differentiating this with respect to x_{132} and x_{322} produces the following simultaneous linear equations.

$$\frac{\partial S_E}{\partial x_{132}} = -5400 + 432x_{132} + 108x_{322} = 0$$

$$\frac{\partial S_E}{\partial x_{322}} = -5724 + 108x_{132} + 432x_{322} = 0$$

Solving these simultaneous linear equations yields the missing value estimates $x_{132} = 9.8$ and $x_{322} = 10.8$.

(2) **Usage**

Double precision:

ierr = ASL_d4mwrm (a, lt, m, ist, isn, ipt, ipn, &x1, v, iwk, wk);

Single precision:

ierr = ASL_r4mwrm (a, lt, m, ist, isn, ipt, ipn, &x1, v, iwk, wk);

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	a	{D∗ / R∗}	m1	Input	Observed values (x_β) . (See Note (a).) Here m1 = $\prod_{i=1}^{m} lt[i-1]$
				Output	Observed values (x_β). However, estimates are stored in the elements corresponding to missing values.
2	lt	I*	m	Input	Number of levels of each factor l_i
3	m	I	1	Input	Number of factors m
4	ist	I*	isn×m	Input	Information about combinations of levels for which missing values occurred (See Note (b))
5	isn	I	1	Input	Number of missing values n_s
6	ipt	I*	m2	Input	Number of factors to be confounded (See Note (c)) However, when ipn > 0, m2 = ipn, and when ipn = 0, the argument ipt should be a dummy argument.
7	ipn	I	1	Input	Number of factors to be confounded
8	x1	{D∗ / R∗}	1	Output	Total mean $\bar{x}$
9	v	{D∗ / R∗}	$(2^m+1)\times 5$	Output	Analysis of variance table (For the storage method of the analysis of variance table, see Table 7−6 in Note (c).)
10	iwk	I*	m3	Work	Work area. m3 = $\prod_{i=1}^{m} lt[i-1] + isn$
11	wk	{D∗ / R∗}	m4	Work	Work area. m4 = $\prod_{i=1}^{m}(lt[i-1]+1) + isn^2 + 2\times isn + 1$
12	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) $1 \leq \mathrm{m} \leq 6$

(b) $\mathrm{lt}[i] \geq 1 \quad (i = 1, \cdots, \mathrm{m})$

(c) $0 \leq \mathrm{ipn} \leq 2^{\mathrm{m}} - 2$

(d) $2 \leq \mathrm{ipt}[i] \leq 2^{\mathrm{m}} - 1 \qquad (i = 0, \cdots, \mathrm{m} - 1)$

(e) $1 \leq \mathrm{isn} < \prod_{i=1}^{\mathrm{m}} (\mathrm{lt[i-1]} - 1)$

(f) $1 \leq \mathrm{ist}[i + j * \mathrm{isn}] \leq \mathrm{lt}[j] \quad (i = 0, 1, \cdots, \mathrm{isn} - 1 \ ; \ j = 0, 1, \cdots, \mathrm{m} - 1)$

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
2000	Estimates of missing values could not be calculated.	The missing value data is replaced by the mean of the non-missing value data, and processing continues.
3000	Any of restrictions (a) to (f) was not satisfied.	Processing is aborted.

(6) **Notes**

(a) Multi-way layout data with no repetitions and having m factors $A_1, A_2, \cdots, A_m$ is represented using m subscripts as follows.

$$x_{j_1 j_2 \cdots j_m} \ (i = 1, \cdots, m; j_i = 1, \cdots, l_i)$$

However, in this function, the multi-way layout data is represented using one subscripts as follows, and that data is stored in array a as a real vector (one-dimensional array).

$$x_{j_1 j_2 \cdots j_m} \rightarrow x_\beta$$

Here, β is defined as follows.

$$\beta = j_1 + \sum_{i=2}^{m} (j_i \prod_{k=1}^{i-1} l_k)$$

(b) The combinations of levels of the i-th missing value $(j_1, j_2, \cdots, j_m)$ are represented as a single integer β according to the method described in Note (a), and that value is stored in $\mathrm{ist}[i-1]$).

(c) The factors that are subject to an analysis of variance are numbered as follows.

- One-way layout

Factor number	Factor
1	Total
2	Error

- Two-way layout

Factor number	Factor
1	Total
2	A_1
3	A_2
4	Error

- Three-way layout

Factor number	Factor
1	Total
2	A_1
3	A_2
4	$A_1 \times A_2$
5	A_3
6	$A_1 \times A_3$
7	$A_2 \times A_3$
8	Error

- Four-way layout

Factor number	Factor
1	Total
2	A_1
3	A_2
4	$A_1 \times A_2$
5	A_3
6	$A_1 \times A_3$
7	$A_2 \times A_3$
8	$A_1 \times A_2 \times A_3$
9	A_4
10	$A_1 \times A_4$
11	$A_2 \times A_4$
12	$A_1 \times A_2 \times A_4$
13	$A_3 \times A_4$
14	$A_1 \times A_3 \times A_4$
15	$A_2 \times A_3 \times A_4$
16	Error

- Five-way layout

Factor number	Factor
1	Total
2	A_1
3	A_2
4	$A_1 \times A_2$
5	A_3
6	$A_1 \times A_3$
7	$A_2 \times A_3$
8	$A_1 \times A_2 \times A_3$
9	A_4
10	$A_1 \times A_4$
11	$A_2 \times A_4$
12	$A_1 \times A_2 \times A_4$
13	$A_3 \times A_4$
14	$A_1 \times A_3 \times A_4$
15	$A_2 \times A_3 \times A_4$
16	$A_1 \times A_2 \times A_3 \times A_4$
17	A_5
18	$A_1 \times A_5$
19	$A_2 \times A_5$
20	$A_1 \times A_2 \times A_5$
21	$A_3 \times A_5$
22	$A_1 \times A_3 \times A_5$
23	$A_2 \times A_3 \times A_5$
24	$A_1 \times A_2 \times A_3 \times A_5$
25	$A_4 \times A_5$
26	$A_1 \times A_4 \times A_5$
27	$A_2 \times A_4 \times A_5$
28	$A_1 \times A_2 \times A_4 \times A_5$
29	$A_3 \times A_4 \times A_5$
30	$A_1 \times A_3 \times A_4 \times A_5$
31	$A_2 \times A_3 \times A_4 \times A_5$
32	Error

- Six-way layout

Factor number	Factor	Factor number	Factor
1	Total	33	$A_1 \times A_6$
2	A_1	34	$A_1 \times A_6$
3	A_2	35	$A_2 \times A_6$
4	$A_1 \times A_2$	36	$A_1 \times A_2 \times A_6$
5	A_3	37	$A_3 \times A_6$
6	$A_1 \times A_3$	38	$A_1 \times A_3 \times A_6$
7	$A_2 \times A_3$	39	$A_2 \times A_3 \times A_6$
8	$A_1 \times A_2 \times A_3$	40	$A_1 \times A_2 \times A_3 \times A_6$
9	A_4	41	$A_4 \times A_6$
10	$A_1 \times A_4$	42	$A_1 \times A_4 \times A_6$
11	$A_2 \times A_4$	43	$A_2 \times A_4 \times A_6$
12	$A_1 \times A_2 \times A_4$	44	$A_1 \times A_2 \times A_4 \times A_6$
13	$A_3 \times A_4$	45	$A_3 \times A_4 \times A_6$
14	$A_1 \times A_3 \times A_4$	46	$A_1 \times A_3 \times A_4 \times A_6$
15	$A_2 \times A_3 \times A_4$	47	$A_2 \times A_3 \times A_4 \times A_6$
16	$A_1 \times A_2 \times A_3 \times A_4$	48	$A_1 \times A_2 \times A_3 A_4 \times A_6$
17	A_5	49	$A_5 \times A_6$
18	$A_1 \times A_5$	50	$A_1 \times A_5 \times A_6$
19	$A_2 \times A_5$	51	$A_2 \times A_5 \times A_6$
20	$A_1 \times A_2 \times A_5$	52	$A_1 \times A_2 \times A_5 \times A_6$
21	$A_3 \times A_5$	53	$A_3 \times A_5 \times A_6$
22	$A_1 \times A_3 \times A_5$	54	$A_1 \times A_3 \times A_5 \times A_6$
23	$A_2 \times A_3 \times A_5$	55	$A_2 \times A_3 \times A_5 \times A_6$
24	$A_1 \times A_2 \times A_3 \times A_5$	56	$A_1 \times A_2 \times A_3 \times A_5 \times A_6$
25	$A_4 \times A_5$	57	$A_4 \times A_5 \times A_6$
26	$A_1 \times A_4 \times A_5$	58	$A_1 \times A_4 \times A_5 \times A_6$
27	$A_2 \times A_4 \times A_5$	59	$A_2 \times A_4 \times A_5 \times A_6$
28	$A_1 \times A_2 \times A_4 \times A_5$	60	$A_1 \times A_2 \times A_4 \times A_5 \times A_6$
29	$A_3 \times A_4 \times A_5$	61	$A_3 \times A_4 \times A_5 \times A_6$
30	$A_1 \times A_3 \times A_4 \times A_5$	62	$A_1 \times A_3 \times A_4 \times A_5 \times A_6$
31	$A_2 \times A_3 \times A_4 \times A_5$	63	$A_2 \times A_3 \times A_4 \times A_5 \times A_6$
32	$A_1 \times A_2 \times A_3 \times A_4 \times A_5$	64	Error

(d) The analysis of variance table elements are stored as follows in the array v.

Table 7−6 Analysis of Variance Table Storage Status

Factor number	Variation	Degrees of freedom	Unbiased variance	Variance ratio	Contribution ratio
1	v[0]	v[n1 + 1]	∗	∗	∗
2	v[1]	v[n1 + 1]	v[2 × n1 + 1]	v[3 × n1 + 1]	v[4 × n1 + 1]
3	v[2]	v[n1 + 2]	v[2 × n1 + 2]	v[3 × n1 + 2]	v[4 × n1 + 2]
⋮	⋮	⋮	⋮	⋮	⋮
n1 − 1	v[n1 − 2]	v[2 × n1 − 2]	v[3 × n1 − 2]	v[4 × n1 − 2]	v[5 × n1 − 2]
n1	v[n1 − 1]	v[2 × n1 − 1]	v[3 × n1 − 1]	∗	v[5 × n1 − 1]

Here, n1 = 2^{m}. The absolute value maximum that can be represented is set for array elements v[2 × n1 + 1], v[3 × n1 + 1], v[4 × n1 + 1] and v[4 × n1 − 1], which are elements for which there is no corresponding analysis of variance table item. In addition, for the error, 0.0 is set for the value of each item for the factors specified in array ipt, which are to be confounded.

(7) **Example**

(a) Problem

Perform an analysis of variance for three-way layout data having no repetitions and for which the number of levels for each factor is 3. However, the observation value vector $X = x_\beta$ is given as follows, where the ∗ portions are missing values.

$$\begin{bmatrix} 13 \\ 21 \\ * \\ 18 \\ 29 \\ 40 \\ 23 \\ 37 \\ 51 \\ 21 \\ * \\ 47 \\ 27 \\ 44 \\ 61 \\ * \\ 54 \\ 75 \\ 29 \\ 47 \\ 65 \\ 36 \\ 59 \\ 82 \\ 43 \\ 71 \\ 99 \end{bmatrix}$$

(b) Input data

Observation matrix X, na=2, n=2, m=3, ipn=0, isn=3, lt[0]=3, lt[1]=3, lt[2]=3, ist[0]=1, ist[3]=3, ist[6]=2, ist[1]=2, ist[4]=1, ist[7]=2, ist[2]=3, ist[5]=1 and ist[8]=1.

(c) Main program

```
/*      C interface example for ASL_d4mwrm */

#include <stdio.h>
#include <stdlib.h>
#include <asl.h>

int main()
{
    double *a;
    int *lt;
    int m;
    int *ipt;
    int ipn;
    int *ist;
    int isn;
    double x1;
    double *v;
    double *wk;
    int *iwk;
    int ierr;
    int i,j,m1,nv,m3,iwk1,iwk2;
    FILE *fp;

    fp = fopen( "d4mwrm.dat", "r" );
    if( fp == NULL )
    {
        printf( "file open error\n" );
        return -1;
    }
```

```
printf( "    *** ASL_d4mwrm ***\n" );
printf( "\n     ** Input **\n\n" );

fscanf( fp, "%d", &m );
printf( "\tm=%6d\n", m );

lt = ( int * )malloc((size_t)( sizeof(int) * m ));
if( lt == NULL )
{
    printf( "no enough memory for array lt\n" );
    return -1;
}

printf( "\n\tNumber of level of each factor\n\n" );
m1 = 1;
m3 = 1;
nv = 1;
printf( "\t" );
for( i=0 ; i<m ; i++)
{
    fscanf( fp, "%d", &lt[i] );
    printf( "%6d ",lt[i] );
    m1 *= lt[i];
    m3 *= lt[i]+1;
    nv *= 2;
}
printf( "\n" );

fscanf( fp, "%d", &ipn );
printf( "\n\tipn=%6d\n\n", ipn );

fscanf( fp, "%d", &isn );
printf( "\tisn=%6d\n\n", isn );

ist = ( int * )malloc((size_t)( sizeof(int) * (isn*m) ));
if( ist == NULL )
{
    printf( "no enough memory for array ist\n" );
    return -1;
}

printf( "\tMissed values\n\n");
for( i=0 ; i<isn ; i++){
    printf( "\t   a(" );
    for( j=0 ; j<m-1 ; j++){
        fscanf( fp, "%d", &ist[i+j*isn] );
        printf( "%6d,",ist[i+j*isn] );
    }
    fscanf( fp, "%d", &ist[i+(m-1)*isn] );
    printf( "%6d)\n",ist[i+(m-1)*isn] );
}

a = ( double * )malloc((size_t)( sizeof(double) * m1 ));
if( a == NULL )
{
    printf( "no enough memory for array a\n" );
    return -1;
}

v = ( double * )malloc((size_t)( sizeof(double) * nv * 5 ));
if( v == NULL )
{
    printf( "no enough memory for array v\n" );
    return -1;
}

iwk = ( int * )malloc((size_t)( sizeof(int) * (m1+isn)));
if( iwk == NULL )
{
    printf( "no enough memory for array iwk\n" );
    return -1;
}

wk = ( double * )malloc((size_t)( sizeof(double) * (m3+isn*isn+2*isn)));
if( wk == NULL )
{
    printf( "no enough memory for array wk\n" );
    return -1;
}

/* Array ipt is not used when ipn is equal to 0 */
ipt=NULL;

printf( "\n\tObservations\n\n" );
for( i=0 ; i<m1 ; i++ )
{
    if( i%5 == 0 )
    {
        printf( "\n\t" );
    }
```

```
        fscanf( fp, "%lf", &a[i] );
        printf( "%8.3g", a[i] );
    }
    printf( "\n\n" );

    fclose( fp );

    ierr = ASL_d4mwrm(a, lt, m, ist, isn, ipt, ipn, &x1, v, iwk, wk);

    printf( "\n     ** Output **\n\n" );
    printf( "\tierr = %6d\n\n", ierr );

    printf( "\tEstimated missed values\n\n");
    for( i=0 ; i<isn ; i++){
        printf( "\t   a(" );
        iwk1 = 0;
        iwk2 = 1;
        for( j=0 ; j<m-1 ; j++){
            printf( "%6d,",ist[i+j*isn] );
            iwk1 += (ist[i+j*isn]-1)*iwk2;
            iwk2 *= lt[j];
        }
        printf( "%6d) = ",ist[i+(m-1)*isn] );
        iwk1 += (ist[i+j*isn]-1)*iwk2;
        iwk2 *= lt[j];
        printf( "%8.3g\n", a[iwk1] );
    }

    printf( "\n\tMean over all levels = %8.3g\n", x1 );

    printf( "\n\n\tAnalysis of variance table\n\n" );
    printf( "\t Factor      S.S.     D.F.     M.S.    V.R.    C.R.\n" );
    printf( "\t-----------------------------------------------------\n" );
    printf( "\t Total  %8.3g %8.3g\n", v[0],v[nv] );
    for( i=1 ; i<nv-1 ; i++)
    {
        printf( "\t %6d %8.3g %8.3g %8.3g %8.3g %8.3g\n",
         i+1,v[i],v[nv+i],v[2*nv+i],v[3*nv+i],v[4*nv+i]);
    }
    printf( "\t Error  %8.3g %8.3g %8.3g          %8.3g\n",
    v[nv-1],v[2*nv-1],v[3*nv-1],v[5*nv-1]);

    free( lt );
    free( ist );
    free( a );
    free( v );
    free( iwk );
    free( wk );

    return 0;
}
```

(d) Output results

```
*** ASL_d4mwrm ***

 ** Input **

m=     3

Number of level of each factor

     3      3      3

ipn=     0

isn=     3

Missed values

   a(     1,     3,     2)
   a(     2,     1,     2)
   a(     3,     1,     1)

Observations


      13      21       1      18      29
      40      23      37      51      21
       1      47      27      44      61
       1      54      75      29      47
      65      36      59      82      43
      71      99


 ** Output **

ierr =      0

Estimated missed values
```

```
   a(     1,     3,     2) =     32.3
   a(     2,     1,     2) =     35.1
   a(     3,     1,     1) =     25.3

Mean over all levels =     43.9


Analysis of variance table

 Factor      S.S.     D.F.     M.S.    V.R.    C.R.
-------------------------------------------------------
 Total  1.19e+04       23
      2  5.1e+03        2 2.55e+03    3e+03     0.43
      3 1.84e+03        2      919 1.08e+03    0.155
      4       231       4     57.8       68   0.0192
      5 4.16e+03        2 2.08e+03 2.45e+03    0.351
      6       479       4      120      141   0.0402
      7      36.3       4     9.08     10.7  0.00278
 Error      4.25        5     0.85           0.00165
```

7.5 CUMULATIVE METHOD

7.5.1 ASL_d4mu01, ASL_r4mu01
Analysis of Variance Using the Cumulative Method

(1) **Function**

The ASL_d4mu01 or ASL_r4mu01 performs an analysis of variance using the cumulative method for data for which the number of repetitions is fixed and the number of factors is at most 3. The number of decision makers is assumed to be r people, and the decision is assumed to be made in d steps. Although the explanation presented below takes as an example the case when the number of factors is 3, one- and two-way layouts can also be handled by setting the number of levels of unused factors to 1.

Let the input data be represented as follows

$$x_{ijkl}^{(s)} \quad \left(\begin{array}{lcll} i & = & 1, \cdots, m_a & ; \text{ Factor A} \\ j & = & 1, \cdots, m_b & ; \text{ Factor B} \\ k & = & 1, \cdots, m_c & ; \text{ Factor C} \\ l & = & 1, \cdots, r & ; \text{ Decision maker R} \\ s & = & 1, \cdots, n & ; \text{ Number of repetitions} \end{array} \right)$$

and let it take integer values from 1 to d. m_a, m_b and m_c are the number of levels in factors A, B, and C, respectively, and r is the number of decision makers. In the explanations below, unless explicitly stated otherwise, summations by the $\sum$ symbol and their weights W_m $(m = 1, \cdots, d-1)$ are defined as follows for subscripts such as i, j, k, l and s.

$$\begin{aligned} \delta_{ijkl}^{(s)(m)} &= \begin{cases} 1 & (\text{When } x_{ijkl}^{(s)} \leq m) \\ 0 & (\text{otherwise}) \end{cases} \\ P_m &= \frac{\sum_i \sum_j \sum_k \sum_l \sum_s \delta_{ijkl}^{(s)(m)}}{m_a \cdot m_b \cdot m_c \cdot r \cdot n} \\ W_m &= \frac{1}{P_m(1 - P_m)} \end{aligned}$$

The correction factor CF is defined as follows.

$$CF = \frac{1}{m_a \cdot m_b \cdot m_c \cdot r \cdot n} \sum_{m=1}^{d-1} \left\{ W_m \sum_i \sum_j \sum_k \sum_l \sum_s \left(\delta_{ijkl}^{(s)(m)} \right)^2 \right\}$$

Also, the analysis of variance table consists of the following elements.

- Sum of squares

$$\begin{aligned} S_{A_i}^{(m)} &= \sum_j \sum_k \sum_l \sum_s \delta_{ijkl}^{(s)(m)} \\ S_{B_j}^{(m)} &= \sum_i \sum_k \sum_l \sum_s \delta_{ijkl}^{(s)(m)} \\ S_{C_k}^{(m)} &= \sum_i \sum_j \sum_l \sum_s \delta_{ijkl}^{(s)(m)} \\ S_{R_l}^{(m)} &= \sum_i \sum_j \sum_k \sum_s \delta_{ijkl}^{(s)(m)} \end{aligned}$$

$$
\begin{aligned}
S^{(m)}_{AB_{ij}} &= \sum_k \sum_l \sum_s \delta^{(s)(m)}_{ijkl} \\
S^{(m)}_{AC_{ik}} &= \sum_j \sum_l \sum_s \delta^{(s)(m)}_{ijkl} \\
S^{(m)}_{BC_{jk}} &= \sum_i \sum_l \sum_s \delta^{(s)(m)}_{ijkl} \\
S^{(m)}_{AR_{il}} &= \sum_j \sum_k \sum_s \delta^{(s)(m)}_{ijkl} \\
S^{(m)}_{BR_{jl}} &= \sum_i \sum_k \sum_s \delta^{(s)(m)}_{ijkl} \\
S^{(m)}_{CR_{kl}} &= \sum_i \sum_j \sum_s \delta^{(s)(m)}_{ijkl} \\
S^{(m)}_{ABC_{ijk}} &= \sum_l \sum_s \delta^{(s)(m)}_{ijkl} \\
&\quad (i = 1, \cdots, m_a; j = 1, \cdots, m_b; k = 1, \cdots, m_c) \\
&\quad (l = 1, \cdots, r; m = 1, \cdots, d-1)
\end{aligned}
$$

$$
\begin{aligned}
S_T &= m_a \cdot m_b \cdot m_c \cdot r \cdot n \cdot (d-1) \\
S_A &= \frac{1}{m_b \cdot m_c \cdot r \cdot n} \sum_{m=1}^{d-1} \sum_i (S^{(m)}_{A_i})^2 W_m - CF \\
S_B &= \frac{1}{m_a \cdot m_c \cdot r \cdot n} \sum_{m=1}^{d-1} \sum_j (S^{(m)}_{B_j})^2 W_m - CF \\
S_C &= \frac{1}{m_a \cdot m_b \cdot r \cdot n} \sum_{m=1}^{d-1} \sum_k (S^{(m)}_{C_k})^2 W_m - CF \\
S_{AB} &= \frac{1}{m_c \cdot r \cdot n} \sum_{m=1}^{d-1} \sum_i \sum_j (S^{(m)}_{AB_{ij}})^2 W_m - CF - S_A - S_B \\
S_{BC} &= \frac{1}{m_a \cdot r \cdot n} \sum_{m=1}^{d-1} \sum_j \sum_k (S^{(m)}_{AB_{ij}})^2 W_m - CF - S_B - S_C \\
S_{AC} &= \frac{1}{m_b \cdot r \cdot n} \sum_{m=1}^{d-1} \sum_i \sum_k (S^{(m)}_{AB_{ij}})^2 W_m - CF - S_A - S_C \\
S_R &= \frac{1}{m_a \cdot m_b \cdot m_c \cdot n} \sum_{m=1}^{d-1} \sum_l (S^{(m)}_{R_l})^2 W_m - CF \\
S_{AR} &= \frac{1}{m_b \cdot m_c \cdot n} \sum_{m=1}^{d-1} \sum_i \sum_l (S^{(m)}_{AR_{il}})^2 W_m - CF - S_A - S_R \\
S_{BR} &= \frac{1}{m_a \cdot m_c \cdot n} \sum_{m=1}^{d-1} \sum_j \sum_l (S^{(m)}_{BR_{jl}})^2 W_m - CF - S_B - S_R \\
S_{CR} &= \frac{1}{m_a \cdot m_b \cdot n} \sum_{m=1}^{d-1} \sum_k \sum_l (S^{(m)}_{CR_{kl}})^2 W_m - CF - S_C - S_R \\
S_{ABC} &= \frac{1}{\cdot r \cdot n} \sum_{m=1}^{d-1} \sum_i \sum_j \sum_k (S^{(m)}_{ABC_{ijk}})^2 W_m - CF - S_A - S_B - S_C - S_{AB} \\
&\quad - S_{AC} - S_{BC}
\end{aligned}
$$

$$\begin{aligned} S_E &= S_T - S_A - S_B - S_C - S_{AB} - S_{AC} - S_{BC} - S_R - S_{AR} - S_{BR} \\ &\quad -S_{CR} - S_{ABC} \end{aligned}$$

- Degrees of freedom

$$\begin{aligned} \phi_T &= (m_a \cdot m_b \cdot m_c \cdot r \cdot n - 1)(d-1) \\ \phi_A &= (m_a - 1)(d-1) \\ \phi_B &= (m_b - 1)(d-1) \\ \phi_C &= (m_c - 1)(d-1) \\ \phi_{AB} &= (m_a - 1)(m_b - 1)(d-1) \\ \phi_{BC} &= (m_b - 1)(m_c - 1)(d-1) \\ \phi_{AC} &= (m_a - 1)(m_c - 1)(d-1) \\ \phi_R &= (r-1)(d-1) \\ \phi_{AR} &= (m_a - 1)(r-1)(d-1) \\ \phi_{BR} &= (m_b - 1)(r-1)(d-1) \\ \phi_{CR} &= (m_c - 1)(r-1)(d-1) \\ \phi_{ABC} &= (m_a - 1)(m_b - 1)(m_c - 1)(d-1) \\ \phi_e &= \phi_T - \phi_A - \phi_B - \phi_C - \phi_{AB} - \phi_{BC} - \phi_{AC} - \phi_R - \phi_{AR} - \phi_{BR} \\ &\quad -\phi_{CR} - \phi_{ABC} \end{aligned}$$

- Unbiased variance

$$\begin{aligned} V_A &= \frac{S_A}{\phi_A} \\ V_B &= \frac{S_B}{\phi_B} \\ V_C &= \frac{S_C}{\phi_C} \\ V_{AB} &= \frac{S_{AB}}{\phi_{AB}} \\ V_{BC} &= \frac{S_{BC}}{\phi_{BC}} \\ V_{AC} &= \frac{S_{AC}}{\phi_{AC}} \\ V_R &= \frac{S_R}{\phi_R} \\ V_{AR} &= \frac{S_{AR}}{\phi_{AR}} \\ V_{BR} &= \frac{S_{BR}}{\phi_{BR}} \\ V_{CR} &= \frac{S_{CR}}{\phi_{CR}} \\ V_{ABC} &= \frac{S_{ABC}}{\phi_{ABC}} \\ V_E &= \frac{S_E}{\phi_E} \end{aligned}$$

- Variance ratio

$$
\begin{aligned}
F_A &= \frac{V_A}{V_E} \\
F_B &= \frac{V_B}{V_E} \\
F_C &= \frac{V_C}{V_E} \\
F_{AB} &= \frac{V_{AB}}{V_E} \\
F_{BC} &= \frac{V_{BC}}{V_E} \\
F_{AC} &= \frac{V_{AC}}{V_E} \\
F_R &= \frac{V_R}{V_E} \\
F_{AR} &= \frac{V_{AR}}{V_E} \\
F_{BR} &= \frac{V_{BR}}{V_E} \\
F_{CR} &= \frac{V_{CR}}{V_E} \\
F_{ABC} &= \frac{V_{ABC}}{V_E}
\end{aligned}
$$

- Contribution ratio

$$
\begin{aligned}
\rho_A &= \frac{S_A - \phi_A \cdot V_E}{S_T} \\
\rho_B &= \frac{S_B - \phi_B \cdot V_E}{S_T} \\
\rho_C &= \frac{S_C - \phi_C \cdot V_E}{S_T} \\
\rho_{AB} &= \frac{S_{AB} - \phi_{AB} \cdot V_E}{S_T} \\
\rho_{BC} &= \frac{S_{BC} - \phi_{BC} \cdot V_E}{S_T} \\
\rho_{AC} &= \frac{S_{AC} - \phi_{AC} \cdot V_E}{S_T} \\
\rho_R &= \frac{S_R - \phi_R \cdot V_E}{S_T} \\
\rho_{AR} &= \frac{S_{AR} - \phi_{AR} \cdot V_E}{S_T} \\
\rho_{BR} &= \frac{S_{BR} - \phi_{BR} \cdot V_E}{S_T} \\
\rho_{CR} &= \frac{S_{CR} - \phi_{CR} \cdot V_E}{S_T} \\
\rho_{ABC} &= \frac{S_{ABC} - \phi_{ABC} \cdot V_E}{S_T}
\end{aligned}
$$

The contribution ratio is calculated only when a significance of 5% or 1% is obtained by an F test. Otherwise, 0.0 is set.

The density frequency is defined as follows.

$$\delta_{ijkl}^{'(s)(m)} = \begin{cases} 1 & \text{When } x_{ijkl}^{(m)} = m \\ 0 & \text{otherwise} \end{cases}$$

$$T_{....} = \sum_i \sum_j \sum_k \sum_l \sum_{m=1}^{n} \delta_{ijkl}^{'(s)(m)}$$

$$T_{i...} = \sum_j \sum_k \sum_l \sum_{m=1}^{n} \delta_{ijkl}^{'(s)(m)}$$

$$T_{.j..} = \sum_i \sum_k \sum_l \sum_{m=1}^{n} \delta_{ijkl}^{'(s)(m)}$$

$$T_{..k.} = \sum_i \sum_j \sum_l \sum_{m=1}^{n} \delta_{ijkl}^{'(s)(m)}$$

$$T_{ij..} = \sum_k \sum_l \sum_{m=1}^{n} \delta_{ijkl}^{'(s)(m)}$$

$$T_{.jk.} = \sum_i \sum_l \sum_{m=1}^{n} \delta_{ijkl}^{'(s)(m)}$$

$$T_{i.k.} = \sum_i \sum_k \sum_{m=1}^{n} \delta_{ijkl}^{'(s)(m)}$$

$$T_{...l} = \sum_i \sum_j \sum_k \sum_{m=1}^{n} \delta_{ijkl}^{'(s)(m)}$$

$$T_{i..l} = \sum_j \sum_k \sum_{m=1}^{n} \delta_{ijkl}^{'(s)(m)}$$

$$T_{.j.l} = \sum_i \sum_k \sum_{m=1}^{n} \delta_{ijkl}^{'(s)(m)}$$

$$T_{..kl} = \sum_i \sum_j \sum_{m=1}^{n} \delta_{ijkl}^{'(s)(m)}$$

$$T_{ij.l} = \sum_k \sum_{m=1}^{n} \delta_{ijkl}^{'(s)(m)}$$

$$T_{.jkl} = \sum_i \sum_{m=1}^{n} \delta_{ijkl}^{'(s)(m)}$$

$$T_{i.kl} = \sum_j \sum_{m=1}^{n} \delta_{ijkl}^{'(s)(m)}$$

$$T_{ijk.} = \sum_l \sum_{m=1}^{n} \delta_{ijkl}^{'(s)(m)}$$

$$T_{ijkl} = \sum_{m=1}^{n} \delta_{ijkl}^{'(s)(m)}$$

The predicted frequency of each level of each factor is defined as follows.

$$\begin{aligned}
\bar{A}_i^{(m)} &= \frac{S_{A_i}^{(m)}}{\sum_{m=1}^{m_a} \bar{A}_i^{(m)}} \\
\bar{B}_j^{(m)} &= \frac{S_{B_j}^{(m)}}{\sum_{m=1}^{m_b} \bar{B}_j^{(m)}} \\
\bar{C}_k^{(m)} &= \frac{S_{C_k}^{(m)}}{\sum_{m=1}^{m_c} \bar{C}_k^{(m)}} \\
\bar{T}^{(m)} &= \frac{\sum_{ijkls} \delta_{ijkl}^{(s)(m)}}{m_a \cdot m_b \cdot m_c \cdot r \cdot n}
\end{aligned}$$

The cumulative predicted frequency $\hat{\mu}_{ijk}^{(m)}$ is defined as follows.

$$\begin{aligned}
\Omega_{ijk}^{(m)} &= \frac{\left(\frac{1}{\bar{A}_i^{(m)}} - 1\right)\left(\frac{1}{\bar{B}_j^{(m)}} - 1\right)\left(\frac{1}{\bar{C}_k^{(m)}} - 1\right)}{\left(\frac{1}{\bar{T}^{(m)}} - 1\right)} \\
\hat{\mu}_{ijk}^{(m)} &= \frac{1}{1 + \Omega_{ijk}^{(m)}} \qquad (m = 1, \cdots, d-1) \\
\hat{\mu}_{ijk}^{(d)} &= 1
\end{aligned}$$

The predicted frequency $\hat{\alpha}_{ijk}^{(m)}$ is defined as follows.

$$\begin{aligned}
\hat{\alpha}_{ijk}^{(1)} &= \hat{\mu}_{ijk}^{(1)} \\
\hat{\alpha}_{ijk}^{(m)} &= \hat{\mu}_{ijk}^{(m)} - \hat{\mu}_{ijk}^{(m-1)} \qquad (m = 2, \cdots, d)
\end{aligned}$$

(2) **Usage**

Double precision:

ierr = ASL_d4mu01 (ia, id, v, ix, nx, &ntc, nt, f, tx, om, ma, am, al, mt, p, g);

Single precision:

ierr = ASL_r4mu01 (ia, id, v, ix, nx, &ntc, nt, f, tx, om, ma, am, al, mt, p, g);

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	ia	I*	m1	Input	Input data $x_{ijkl}^{(s)}$ (See Note (a)) Here, m1 = $\prod_{i=0}^{4}$ id[i]
2	id	I*	6	Input	Number of levels of each factor, number of decision makers, number of repetitions, and number of steps (See Note (b))
3	v	{D∗ / R∗}	m2	Output	Analysis of variance table (For the storage method of the analysis of variance table, see Tables 7−7, 7−8 and 7−9 in note (c).) Here, m2 is 28 when the number of factors is 1, 46 when the number of factors is 2, and 74 when the number of factors is 3.
4	ix	I*	nx×id[5]	Output	Density frequency table (See Note (e))
5	nx	I	1	Input	Adjustable dimension of arrays ix and nt
6	ntc	I*	1	Output	Size of density frequency table
7	nt	I*	nx×4	Output	Frequency density table numbers (See Note (e))
8	f	{D∗ / R∗}	3×9×id[5]	Output	Frequency in each step of each level $\bar{A}_i^{(m)}$, $\bar{B}_j^{(m)}$, $\bar{C}_k^{(m)}$
9	tx	{D∗ / R∗}	id[5]	Output	$\bar{T}^{(m)}$
10	om	{D∗ / R∗}	ma×id[5]	Output	$\Omega_{ijk}^{(m)}$ (See Note (f))
11	ma	I	1	Input	Adjustable dimension of arrays om, am, al and mt
12	am	{D∗ / R∗}	ma×id[5]	Output	Predicted cumulative frequency $\hat{\mu}_{ijk}^{(m)}$ (See Note (f))
13	al	{D∗ / R∗}	ma×id[5]	Output	Predicted frequency $\hat{\alpha}_{ijk}^{(m)}$(See Note (f))
14	mt	I*	ma×3	Output	$\Omega_{ijk}^{(m)}$, $\hat{\mu}_{ijk}^{(m)}$ or $\hat{\alpha}_{ijk}^{(m)}$ numbers (See Note (f))

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
15	p	$\left\{\begin{matrix}\text{D}*\\ \text{R}*\end{matrix}\right\}$	id[5]	Output	P_m
16	g	$\left\{\begin{matrix}\text{D}*\\ \text{R}*\end{matrix}\right\}$	id[5]	Output	Weight W_m
17	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) $2 \leq \text{id[0]} \leq 9$

(b) $1 \leq \text{id[i]} \leq 9 \quad (\text{i} = 1, 2)$

(c) $1 \leq \text{id[3]} \leq 100$

(d) $1 \leq \text{id[4]} \leq 5$

(e) $1 \leq \text{id[5]} \leq 11$

(f) $\text{nx} \geq (\text{id[3]} + 1) \times \prod_{\text{i}=0,1,2,\text{id[i]}\neq 1} (\text{id[i]} + 1)$

(g) $\text{ma} \geq \prod_{\text{i}=0}^{2} \text{id[i]}$

(h) If the number of factors of the input data is 2, id[2] = 1. If the number of factors of the input data is 1, id[1] = id[2] = 1.

(i) $1 \leq \text{ia[i]} \leq \text{id[5]}$
$(\text{i} = 0, 1, \cdots, \prod_{\text{j}=0}^{4} \text{id[j]} - 1)$

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
3000	Any of restriction (a) to (g) was not satisfied.	Processing is aborted.
3010	Restriction (h) was not satisfied.	
3020	Restriction (i) was not satisfied.	

(6) **Notes**

(a) The input data $x_{ijkl}^{(s)}$ is stored in array ia as follows.

$$\begin{aligned}\text{ia}[\alpha] &= x_{ijkl}^{(s)} \\ \alpha &= i - 1 + (j-1) \times \text{id[0]} \\ &\quad + (k-1) \times \text{id[0]} \times \text{id[1]} \\ &\quad + (l-1) \times \text{id[0]} \times \text{id[1]} \times \text{id[2]} \\ &\quad + (s-1) \times \text{id[0]} \times \text{id[1]} \times \text{id[2]} \times \text{id[3]}\end{aligned}$$

That is, the input data is stored so that the subscripts vary in the order of i, j, k and l.

(b) Information related to the input data $x_{ijkl}^{(s)}$ is stored in array id as follows.

id[0] : Number of levels of factor A m_a
id[1] : Number of levels of factor B m_b
id[2] : Number of levels of factor C m_c
id[3] : Number of decision makers r
id[4] : Number of repetitions n
id[5] : Number of steps d

If the number of factors is 2, the number of levels of factor C should be set to 1. If the number of factors is 1, the number of levels of factor B and the number of levels of factor C should be set to 1.

(c) The analysis of variance table elements are stored as follows in the array v.

- Number of factors = 1

Table 7−7 Analysis of Variance Table Storage Status (Number of Factors = 1)

Factor	Sum of squares	Degrees of freedom	Unbiased variance	Variance ratio	Contribution ratio	F test result
T	v[0]	v[6]				
A	v[1]	v[7]	v[12]	v[17]	v[20]	v[25]
R	v[2]	v[8]	v[13]	v[18]	v[21]	v[26]
AR	v[3]	v[9]	v[14]	v[19]	v[22]	v[27]
E	v[4]	v[10]	v[15]		v[23]	
(E)	v[5]	v[11]	v[16]		v[24]	

- Number of factors = 2

Table 7−8 Analysis of Variance Table Storage Status (Number of Factors = 2)

Factor	Sum of squares	Degrees of freedom	Unbiased variance	Variance ratio	Contribution ratio	F test result
T	v[0]	v[9]				
A	v[1]	v[10]	v[18]	v[26]	v[32]	v[40]
B	v[2]	v[11]	v[19]	v[27]	v[33]	v[41]
R	v[3]	v[12]	v[20]	v[28]	v[34]	v[42]
AB	v[4]	v[13]	v[21]	v[29]	v[35]	v[43]
AR	v[5]	v[14]	v[22]	v[30]	v[36]	v[44]
BR	v[6]	v[15]	v[23]	v[31]	v[37]	v[45]
E	v[7]	v[16]	v[24]		v[38]	
(E)	v[8]	v[17]	v[25]		v[39]	

- Number of factors = 3

(d) The value of an F test result term of array v is set to 1.0 if the test result is a 1% significance, and it is set to 5.0 if the test result is a 5% significance. Otherwise, it is set to 0.0.

(e) The method of storing the density frequency table in array ix is explained below by taking as an example the case when the number of factors is 3. First, the following subscripts are provided corresponding to the subscripts i, j, k and l of density frequency $T_{ijkl}^{(m)}$.

$$i' = 0, 1, 2, \cdots, m_a$$
$$j' = 0, 1, 2, \cdots, m_b$$

Table 7−9 Analysis of Variance Table Storage Status (Number of Factors = 3)

Factor	Sum of squares	Degrees of freedom	Unbiased variance	Variance ratio	Contribution ratio	F test result
T	v[0]	v[12]				
A	v[1]	v[13]	v[26]	v[39]	v[50]	v[63]
B	v[2]	v[14]	v[27]	v[40]	v[51]	v[64]
C	v[3]	v[15]	v[28]	v[41]	v[52]	v[65]
R	v[4]	v[16]	v[29]	v[42]	v[53]	v[66]
AB	v[4]	v[17]	v[30]	v[43]	v[54]	v[67]
AC	v[5]	v[18]	v[31]	v[44]	v[55]	v[68]
AR	v[6]	v[19]	v[32]	v[45]	v[56]	v[69]
BC	v[7]	v[20]	v[33]	v[46]	v[57]	v[70]
BR	v[7]	v[21]	v[34]	v[47]	v[58]	v[71]
CR	v[8]	v[22]	v[35]	v[48]	v[59]	v[72]
ABC	v[9]	v[23]	v[36]	v[49]	v[60]	v[73]
E	v[10]	v[24]	v[37]		v[61]	
(E)	v[11]	v[25]	v[38]		v[62]	

$$k' = 0, 1, 2, \cdots, m_c$$
$$l' = 0, 1, 2, \cdots, r$$

Then, $D^{(m)}_{i'j'k'l'}$ is defined as follows.
When none of i', j', k' and l' is 0 :

$$D^{(m)}_{i'j'k'l'} = T^{(m)}_{i'j'k'l'}$$

When any of i', j', k' and l' is 0 :
$D_{i'j'k'l'}$ is defined as the summation of $T^{(m)}_{i'j'k'l'}$ for the subscript or subscripts that are 0. For example, when $j' = 0$ and i', k' and l' are not 0, $D^{(m)}_{i'j'k'l'}$ is defined as follows.

$$D^{(m)}_{i'j'k'l'} = \sum_j T^{(m)}_{i'jk'l'} = T^{(m)}_{i'\cdot k'l'}$$

Or, when $i' = k' = 0$ and j' and l' are not 0, $D^{(m)}_{i'j'k'l'}$ is defined as follows.

$$D^{(m)}_{i'j'k'l'} = \sum_i \sum_k T^{(m)}_{ij'kl'} = T^{(m)}_{\cdot j'\cdot l'}$$

The density frequency table and these numbers are stored as follows in arrays ix and nt.

$$\mathrm{ix}[\beta + (m-1) \times nd] = D^{(m)}_{i'j'k'l'}$$

$$\mathrm{nt}[\beta] = i'$$

$$\mathrm{nt}[\beta + nx] = j'$$

$$\mathrm{nt}[\beta + 2 \times nx] = k'$$

$$\mathrm{nt}[\beta + 3 \times nx] = l'$$

$$\beta = l' + k' \times \mathrm{id}[3] + j' \times \mathrm{id}[2] \times \mathrm{id}[3] + i' \times \mathrm{id}[1] \times \mathrm{id}[2] \times \mathrm{id}[3]$$

That is, the subscripts of $T_{i'j'k'l'}$ vary in the order of l', k', j' and i' according to the variation of array subscript β. However, if the number of factors is 2, the values of nt$[\beta + 2 \times \mathrm{nx}]$ are set to 0, and if the number of factors is 1, the values of nt$[\beta + \mathrm{nx}]$ and nt$[\beta + 2 \times \mathrm{nx}]$ are set to 0.

(f) $\Omega_{ijk}^{(m)}$, $\hat{\mu}_{ijk}^{(m)}$, and $\hat{\alpha}_{ijk}^{(m)}$ and the numbers corresponding to them are stored as follows in arrays om, am, al and mt.

$$\mathrm{om}[\beta + (m-1) \times ma] = \Omega_{ijk}^{(m)}$$

$$\mathrm{am}[\beta + (m-1) \times ma] = \hat{\mu}_{ijk}^{(m)}$$

$$\mathrm{al}[\beta + (m-1) \times ma] = \hat{\alpha}_{ijk}^{(m)}$$

$$\mathrm{mt}[\beta] = i$$

$$\mathrm{mt}[\beta + ma] = j$$

$$\mathrm{mt}[\beta + 2 \times ma] = k$$

$$\beta = k + j \times \mathrm{id}[2] + i \times \mathrm{id}[1] \times \mathrm{id}[2]$$

That is, the subscripts of $\Omega_{ijk}^{(m)}$, $\hat{\mu}_{ijk}^{(m)}$, and $\hat{\alpha}_{ijk}^{(m)}$ vary in the order of k, j and i according to the variation of array subscript β. However, if the number of factors is 2, the values of mt$[\beta + 2 \times \mathrm{ma}]$ are set to 0, and if the number of factors is 1, the values of mt$[\beta + \mathrm{ma}]$ and mt$[\beta + 2 \times \mathrm{ma}]$ are set to 0.

(7) **Example**

(a) Problem

Perform an analysis of variance according to the cumulative method for the following observation data for which the number of factors is 2.

$$\begin{aligned} X \quad = \quad & \{2,3,3,1,2,3,2,3,2,1,2,2,2,3,2,1,1,3,2,3, \\ & 3,1,3,3,2,3,2,1,3,2,2,3,3,1,2,3,3,2,3,2, \\ & 3,2,3,2,3,1,2,1,3,1,3,1,2,3,1,2,3,1,3,3, \\ & 2,3,3,2,3,2,1,3,2,2,2,3\} \end{aligned}$$

Assume that the number of levels of factor A is 3, the number of levels of factor B is 2, the number of decision makers is 6, the number of repetitions is 2 and that the decisions are made in 3 steps.

(b) Input data

Observed data X, nx=84, ma=6,
id[0]=3, id[1]=2, id[2]=1, id[3]=6, id[4]=2 and id[5]=3.

(c) Main program

```
/*      C interface example for ASL_d4mu01 */

#include <stdio.h>
#include <stdlib.h>
#include <asl.h>

int main()
{
    int *ia, id[6];
    double *v;
    int *ix;
    int nx, ntc;
    int *nt;
    double *f, *tx, *om;
    int ma;
    double *am, *al;
    int *mt;
    double *p, *g;
    int ierr;

    int i,j,m1,m2;
    FILE *fp;

    fp = fopen( "d4mu01.dat", "r" );
    if( fp == NULL )
```

```
{
    printf( "file open error\n" );
    return -1;
}

printf( "    *** ASL_d4mu01 ***\n" );
printf( "\n     ** Input **\n\n" );

fscanf( fp, "%d", &nx );
fscanf( fp, "%d", &ma );

for( i=0 ; i<6 ; i++ )
{
    fscanf( fp, "%d", &id[i] );
}
printf( "\tNumber of A         (id[0]) = %3d\n" , id[0] );
printf( "\tNumber of B         (id[1]) = %3d\n" , id[1] );
printf( "\tNumber of C         (id[2]) = %3d\n" , id[2] );
printf( "\tNumber of persons   (id[3]) = %3d\n" , id[3] );
printf( "\tNumber of iterations(id[4]) = %3d\n" , id[4] );
printf( "\tNumber of steps     (id[5]) = %3d\n\n" , id[5] );

m1 = id[0]*id[1]*id[2]*id[3]*id[4];
ia = ( int * )malloc((size_t)( sizeof(int) * m1 ));
if( ia == NULL )
{
    printf( "no enough memory for array ia\n" );
    return -1;
}

v = ( double * )malloc((size_t)( sizeof(double) * 46 ));
if( v == NULL )
{
    printf( "no enough memory for array v\n" );
    return -1;
}

ix = ( int * )malloc((size_t)( sizeof(int) * (nx*id[5]) ));
if( ix == NULL )
{
    printf( "no enough memory for array ix\n" );
    return -1;
}

nt = ( int * )malloc((size_t)( sizeof(int) * (nx*4) ));
if( nt == NULL )
{
    printf( "no enough memory for array nt\n" );
    return -1;
}

f = ( double * )malloc((size_t)( sizeof(double) * (3*9*id[5]) ));
if( f == NULL )
{
    printf( "no enough memory for array f\n" );
    return -1;
}

tx = ( double * )malloc((size_t)( sizeof(double) * id[5] ));
if( tx == NULL )
{
    printf( "no enough memory for array tx\n" );
    return -1;
}

om = ( double * )malloc((size_t)( sizeof(double) * (ma*id[5]) ));
if( om == NULL )
{
    printf( "no enough memory for array om\n" );
    return -1;
}

am = ( double * )malloc((size_t)( sizeof(double) * (ma*id[5]) ));
if( am == NULL )
{
    printf( "no enough memory for array am\n" );
    return -1;
}

al = ( double * )malloc((size_t)( sizeof(double) * (ma*id[5]) ));
if( al == NULL )
{
    printf( "no enough memory for array al\n" );
    return -1;
}

mt = ( int * )malloc((size_t)( sizeof(int) * (ma*3) ));
if( mt == NULL )
{
    printf( "no enough memory for array mt\n" );
    return -1;
}
```

```
p = ( double * )malloc((size_t)( sizeof(double) * id[5] ));
if( p == NULL )
{
    printf( "no enough memory for array p\n" );
    return -1;
}

g = ( double * )malloc((size_t)( sizeof(double) * id[5] ));
if( g == NULL )
{
    printf( "no enough memory for array g\n" );
    return -1;
}

printf( "\tnx = %4d\n", nx );
printf( "\tma = %4d\n\n", ma );

printf( "\tObservations\n" );
for( i=0 ; i<m1 ; i++ )
{
    if( i%10 == 0 )
    {
        printf( "\n\t" );
    }
    fscanf( fp, "%d", &ia[i] );
    printf( "%5d", ia[i] );
}

fclose( fp );

ierr = ASL_d4mu01(ia, id, v, ix, nx, &ntc, nt, f,
         tx, om, ma, am, al, mt, p, g);

printf( "\n\n     ** Output **\n\n" );
printf( "\tierr = %6d\n", ierr );

printf( "\n\tAnalysis of variance table\n\n" );
printf( "\t Factor      S.S.     D.F.     M.S." );
printf( "    V.R.    C.R.       R.F.\n" );
printf( "\t-----------------------------------" );
printf( "--------------------------\n" );
printf( "\t      1 %8.3g %8.3g\n", v[0],v[9] );
for( i=2 ; i<8 ; i++)
{
    printf( "\t %6d %8.3g %8.3g %8.3g %8.3g %8.3g %8.3g %8.3g\n",
    i,v[i-1],v[i+8],v[i+16],v[i+24],v[i+30],v[i+38]);
}
printf( "\t      8 %8.3g %8.3g %8.3g          %8.3g\n",
    v[7],v[16],v[24],v[38]);
printf( "\t      9 %8.3g %8.3g %8.3g          %8.3g\n",
    v[8],v[17],v[25],v[39]);

printf( "\n\tDensity frequency\n\n" );
printf( "\t   ABCR\n" );
printf( "\t   -------------------------\n" );
for( i=0 ; i<ntc ; i++ )
{
    printf( "\t   %1d%1d%1d%1d ",
        nt[i],nt[i+nx],nt[i+2*nx],nt[i+3*nx]);
    for( j=0 ; j<id[5] ; j++ )
    {
        printf( "%6d ", ix[i+j*nx]);
    }
    printf( "\n" );
}

printf( "\n\tFrequencies at n step in A level\n\n" );
for( i=0 ; i<id[0] ; i++ )
{
    printf( "\t" );
    for( j=0 ; j<id[5] ; j++ )
    {
        printf( "%8.3g ", f[3*i+27*j] );
    }
    printf( "\n" );
}

printf( "\n\tFrequencies at n step in B level\n\n" );
for( i=0 ; i<id[1] ; i++ )
{
    printf( "\t" );
    for( j=0 ; j<id[5] ; j++ )
    {
        printf( "%8.3g ", f[1+3*i+27*j] );
    }
    printf( "\n" );
}
```

```
    printf( "\n\tFrequencies at n step in tx level\n\n" );
    printf( "\t" );
    for( j=0 ; j<id[5]-1 ; j++ )
    {
        printf( "%8.3g ", tx[j] );
    }
    printf( "\n" );

    m2 = id[0]*id[1]*id[2];
    printf( "\n\tOmega\n\n" );
    printf( "\t   ABC\n" );
    printf( "\t   ----------------------\n" );
    for( i=0 ; i<m2 ; i++ )
    {
        printf( "\t   %1d%1d%1d ",mt[i],mt[i+ma],mt[i+2*ma]);
        for( j=0 ; j<id[5]-1 ; j++ )
        {
            printf( "%8.3g ", om[i+j*ma] );
        }
        printf( "\n" );
    }

    printf( "\n\tMu\n\n" );
    printf( "\t   ABC\n" );
    printf( "\t   ----------------------\n" );
    for( i=0 ; i<m2 ; i++ )
    {
        printf( "\t   %1d%1d%1d ",mt[i],mt[i+ma],mt[i+2*ma]);
        for( j=0 ; j<id[5]-1 ; j++ )
        {
            printf( "%8.3g ", am[i+j*ma] );
        }
        printf( "\n" );
    }

    printf( "\n\tAlpha\n\n" );
    printf( "\t   ABC\n" );
    printf( "\t  ----------------------\n" );
    for( i=0 ; i<m2 ; i++ )
    {
        printf( "\t   %1d%1d%1d ",mt[i],mt[i+ma],mt[i+2*ma]);
        for( j=0 ; j<id[5]-1 ; j++ )
        {
            printf( "%8.3g ", al[i+j*ma] );
        }
        printf( "\n" );
    }

    printf( "\n\tP\n\n" );
    printf( "\t" );
    for( i=0 ; i<id[5] ; i++ )
    {
        printf( "%8.3g ", p[i] );
    }
    printf( "\n" );

    printf( "\n\tWeight\n\n" );
    printf( "\t" );
    for( i=0 ; i<id[5] ; i++ )
    {
        printf( "%8.3g ", g[i] );
    }
    printf( "\n" );

    free( ia );
    free( v );
    free( ix );
    free( nt );
    free( f );
    free( tx );
    free( om );
    free( am );
    free( al );
    free( mt );
    free( p );
    free( g );

    return 0;
}
```

(d) Output results

```
*** ASL_d4mu01 ***

 ** Input **

Number of A         (id[0]) =   3
```

```
Number of B         (id[1]) =   2
Number of C         (id[2]) =   1
Number of persons   (id[3]) =   6
Number of iterations(id[4]) =   2
Number of steps     (id[5]) =   3

nx =   84
ma =    6

Observations

    2    3    3    1    2    3    2    3    2    1
    2    2    2    3    2    1    1    3    2    3
    3    1    3    3    2    3    2    1    3    2
    2    3    3    1    2    3    3    2    3    2
    3    2    3    2    3    1    2    1    3    1
    3    1    2    3    1    2    3    1    3    3
    2    3    3    2    3    2    1    3    2    2
    2    3

 ** Output **

ierr =      0

Analysis of variance table

 Factor      S.S.     D.F.      M.S.     V.R.     C.R.        R.F.
-------------------------------------------------------------------
      1       144      142
      2      30.2        4     7.55     10.8    0.189           1
      3      8.45        2     4.23     6.06   0.0484           1
      4       6.9       10     0.69     0.99        0           0
      5      8.08        4     2.02      2.9   0.0356           5
      6      20.4       20     1.02     1.46        0           0
      7      5.84       10    0.584    0.838        0           0
      8      64.1       92    0.697                0
      9      97.3      132    0.737            0.727

Density frequency

   ABCR
   --------------------------
   0000      14      27      31
   1000      11      10       3
   2000       2       9      13
   3000       1       8      15
   0100       3      14      19
   0200      11      13      12
   0001       1       5       6
   0002       3       6       3
   0003       4       3       5
   0004       3       2       7
   0005       1       6       5
   0006       2       5       5
   1100       2       7       3
   1200       9       3       0
   2100       1       3       8
   2200       1       6       5
   3100       0       4       8
   3200       1       4       7
   1001       1       2       1
   1002       2       1       1
   1003       2       1       1
   1004       3       1       0
   1005       1       3       0
   1006       2       2       0
   2001       0       2       2
   2002       0       3       1
   2003       2       1       1
   2004       0       1       3
   2005       0       0       4
   2006       0       2       2
   3001       0       1       3
   3002       1       2       1
   3003       0       1       3
   3004       0       0       4
   3005       0       3       1
   3006       0       1       3
   0101       0       2       4
   0102       0       3       3
   0103       1       2       3
   0104       1       2       3
   0105       0       3       3
   0106       1       2       3
   0201       1       3       2
   0202       3       3       0
   0203       3       1       2
   0204       2       0       4
   0205       1       3       2
   0206       1       3       2
   1101       0       1       1
   1102       0       1       1
   1103       0       1       1
   1104       1       1       0
```

```
   1105      0      2      0
   1106      1      1      0
   1201      1      1      0
   1202      2      0      0
   1203      2      0      0
   1204      2      0      0
   1205      1      1      0
   1206      1      1      0
   2101      0      1      1
   2102      0      1      1
   2103      1      0      1
   2104      0      1      1
   2105      0      0      2
   2106      0      0      2
   2201      0      1      1
   2202      0      2      0
   2203      1      1      0
   2204      0      0      2
   2205      0      0      2
   2206      0      2      0
   3101      0      0      2
   3102      0      1      1
   3103      0      1      1
   3104      0      0      2
   3105      0      1      1
   3106      0      1      1
   3201      0      1      1
   3202      1      1      0
   3203      0      0      2
   3204      0      0      2
   3205      0      2      0
   3206      0      0      2

Frequencies at n step in A level

   0.458    0.417    0.125
  0.0833    0.375    0.542
  0.0417    0.333    0.625

Frequencies at n step in B level

  0.0833    0.389    0.528
   0.306    0.361    0.333

Frequencies at n step in tx level

   0.194    0.569

Omega

   ABC
   ----------------------
   110     3.14    0.211
   120    0.648   0.0945
   210     29.2     1.75
   220     6.03    0.782
   310     61.1     2.46
   320     12.6      1.1

Mu

   ABC
   ----------------------
   110    0.242    0.826
   120    0.607    0.914
   210   0.0331    0.364
   220    0.142    0.561
   310   0.0161    0.289
   320   0.0734    0.476

Alpha

   ABC
  ----------------------
   110    0.242    0.584
   120    0.607    0.307
   210   0.0331    0.331
   220    0.142    0.419
   310   0.0161    0.273
   320   0.0734    0.402

P

   0.194    0.569        1

Weight

    6.38     4.08        1
```

7.6 RANDOMIZED BLOCK DESIGN

7.6.1 ASL_d4rb01, ASL_r4rb01
Analysis of Variance Using Randomized Block Design

(1) **Function**

The ASL_d4rb01 or ASL_r4rb01 performs an analysis of variance using randomized block design.

The analysis of variance results for observed values $\{x_{ij}\}$, $(i = 1, \cdots, n;\ j = 1, \cdots, t)$ obtained using n blocks and t trials are defined as follows.

Total sum :

$$T = \sum_{i=1}^{n} \sum_{j=1}^{t} x_{ij}$$

Total sum for each block :

$$T_{i\cdot} = \sum_{j=1}^{t} x_{ij}$$

Total sum for each trial :

$$T_{\cdot j} = \sum_{i=1}^{n} x_{ij}$$

Variation :

- Total variation
$$S = \sum_{i=1}^{n} \sum_{j=1}^{t} \left(x_{ij} - \frac{T}{n \cdot t} \right)^2$$
- Mean variation
$$S_c = \frac{T^2}{n \cdot t}$$
- Inter-block variation
$$S_p = \frac{1}{t} \sum_{i=1}^{n} \left(T_{i\cdot} - \frac{T}{n} \right)^2$$
- Inter-trial variation
$$S_r = \frac{1}{n} \sum_{j=1}^{t} \left(T_{\cdot j} - \frac{T}{t} \right)^2$$
- Error variation
$$S_e = S - (S_p + S_r)$$

Degrees of freedom :

$$\phi = n \cdot t - 1, \quad \phi_c = 1, \quad \phi_p = n - 1, \quad \phi_r = t - 1$$

$$\phi_e = (n - 1)(t - 1)$$

Unbiased variance :

$$V_p = \frac{S_p}{\phi_p}, \quad V_r = \frac{S_r}{\phi_r}, \quad V_e = \frac{S_e}{\phi_e}$$

Variance ratio :

$$F_p = \frac{V_p}{V_e}, \quad F_r = \frac{V_r}{V_e}$$

(2) **Usage**

Double precision:

```
ierr = ASL_d4rb01  (a, na, nb, nt, v);
```

Single precision:

```
ierr = ASL_r4rb01  (a, na, nb, nt, v);
```

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	a	{D∗ / R∗}	na×nt	Input	Matrix in which observed values are stored (x_{ij})
2	na	I	1	Input	Adjustable dimension of array a
3	nb	I	1	Input	Number of blocks n
4	nt	I	1	Input	Number of trials t
5	v	{D∗ / R∗}	15	Output	Analysis of variance results (See Table 7−10 in note (a))
6	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) na≥nb≥1

(b) nt≥1

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
1000	nb=1 or nt=1 was specified.	The absolute value maximum that can be represented is set for the unbiased variance and variance ratio.
3000	Restriction (a), (b) was not satisfied.	Processing is aborted.

(6) **Notes**

(a) The analysis of variance results are stored as follows in the array v.

Table 7−10 Analysis of Variance Table

Factor	Variation	Degrees of freedom	Unbiased variance	Variance ratio
Total	v[0]	v[5]		
Mean	v[1]	v[6]		
Inter-block	v[2]	v[7]	v[10]	v[13]
Inter-trial	v[3]	v[8]	v[11]	v[14]
Error	v[4]	v[9]	v[12]	

(7) **Example**

(a) Problem

When the observed values are given by matrix X shown below, perform an analysis of variance using randomized block design.

$$X = \begin{bmatrix} 42 & 28 & 19 & 7 & 4 \\ 15 & 14 & -19 & -4 & -6 \\ -8 & 30 & -17 & 9 & -31 \end{bmatrix}$$

(b) Input data

Observation matrix X, na=100, nb=3 and nt=5.

(c) Main program

```
/*      C interface example for ASL_d4rb01 */

#include <stdio.h>
#include <stdlib.h>
#include <asl.h>

int main()
{
    double *a;
    int na;
    int nb;
    int nt;
    double v[15];
    int ierr;
    int i,j;
    FILE *fp;

    fp = fopen( "d4rb01.dat", "r" );
    if( fp == NULL )
    {
        printf( "file open error\n" );
        return -1;
    }

    printf( "    *** ASL_d4rb01 ***\n" );
    printf( "\n     ** Input **\n\n" );

    fscanf( fp, "%d", &na );
    fscanf( fp, "%d", &nb );
    fscanf( fp, "%d", &nt );

    a = ( double * )malloc((size_t)( sizeof(double) * (na*nt) ));
    if( a == NULL )
    {
        printf( "no enough memory for array a\n" );
        return -1;
    }

    printf( "\tna=%3d,  nb=%3d,  nt=%3d\n", na, nb, nt );

    printf("\n\tObservations\n\n");
    for( i=0 ; i<nb ; i++ )
    {
        printf( "\t" );
        for( j=0 ; j<nt ; j++ )
```

```
        {
            fscanf( fp, "%lf", &a[i+na*j] );
            printf( "%8.3g ", a[i+na*j] );
        }
        printf( "\n" );
    }

    fclose( fp );

    ierr = ASL_d4rb01(a, na, nb, nt, v);

    printf( "\n     ** Output **\n\n" );
    printf( "\tierr = %6d\n", ierr );

    printf( "\n\tAnalysis of variance table\n\n" );
    printf( "\tFactor       S.S.       D.F.     M.S.    V.R.\n" );
    printf( "\t-------------------------------------------\n" );
    printf( "\tTotal     %8.3g %8.3g\n", v[0],v[5] );
    printf( "\tMean      %8.3g %8.3g\n", v[1],v[6]);
    printf( "\tBlock     %8.3g %8.3g %8.3g %8.3g\n",
         v[2],v[7],v[10],v[13]);
    printf( "\tTreatment %8.3g %8.3g %8.3g %8.3g\n",
         v[3],v[8],v[11],v[14]);
    printf( "\tError     %8.3g %8.3g %8.3g\n",
        v[4],v[9],v[12]);

    free( a );

    return 0;
}
```

(d) Output results

```
*** ASL_d4rb01 ***

 ** Input **

na=100,  nb=  3,  nt=  5

Observations

      42       28       19        7        4
      15       14      -19       -4       -6
      -8       30      -17        9      -31

 ** Output **

ierr =      0

Analysis of variance table

Factor       S.S.       D.F.     M.S.    V.R.
-----------------------------------------------
Total     5.64e+03       14
Mean           459        1
Block      1.6e+03        2      799     4.35
Treatment 2.58e+03        4      644     3.51
Error     1.47e+03        8      184
```

7.7 GRECO-LATIN SQUARE METHOD

7.7.1 ASL_d4gl01, ASL_r4gl01
Analysis of Variance Using the Greco-Latin Square Method

(1) **Function**

The ASL_d4gl01 or ASL_r4gl01 performs an analysis of variance using the Greco-Latin Square method. The analysis of variance results for the two orthogonal Latin squares MT and MG and observed values $\{x_{ij(kl)}\}$ (i: Levels of factor P; j: Levels of factor Q; k and l: Constitute Greco-Latin square) are defined as follows.

Total sum, row sum, and column sum :

$$T = \sum_{i=1}^{n}\sum_{j=1}^{n} x_{ij(kl)}, \quad T_{i\cdot} = \sum_{j=1}^{n} x_{ij(kl)}, \quad T_{\cdot j} = \sum_{i=1}^{n} x_{ij(kl)}$$

$$T_k = \sum_{k\in MT} x_{ij(kl)}, \quad T_l = \sum_{l\in MG} x_{ij(kl)}$$

Variation:

- Total variation
$$S_s = \sum_{i=1}^{n}\sum_{j=1}^{n}\left(x_{ij(kl)} - \frac{T}{n^2}\right)^2$$
- Inter-row variation
$$S_p = \frac{1}{n}\sum_{i=1}^{n}\left(T_{i\cdot} - \frac{T}{n}\right)^2$$
- Inter-column variation
$$S_q = \frac{1}{n}\sum_{j=1}^{n}\left(T_{\cdot j} - \frac{T}{n}\right)^2$$
- Inter-R variation
$$S_r = \frac{1}{n}\sum_{k=1}^{n}\left(T_k - \frac{T}{n}\right)^2$$
- Inter-A variation
$$S_a = \frac{1}{n}\sum_{l=1}^{n}\left(T_l - \frac{T}{n}\right)^2$$
- Error variation
$$S_e = S_s - (S_p + S_q + S_r + S_a)$$

Degrees of freedom :

$$\phi_s = n^2 - 1, \quad \phi_p = \phi_q = \phi_r = \phi_a = n - 1, \quad \phi_e = (n-1)(n-3)$$

Unbiased variance :

$$V_p = \frac{S_p}{\phi_p}, \quad V_q = \frac{S_q}{\phi_q}, \quad V_r = \frac{S_r}{\phi_r}, \quad V_a = \frac{S_a}{\phi_a}, \quad V_e = \frac{S_e}{\phi_e}$$

Variance ratio :

$$F_p = \frac{V_p}{V_e}, \quad F_q = \frac{V_q}{V_e}, \quad F_r = \frac{V_r}{V_e}, \quad F_a = \frac{V_a}{V_e}$$

(2) **Usage**

Double precision:

ierr = ASL_d4gl01 (a, na, n, mt, mg, v, iwk, wk);

Single precision:

ierr = ASL_r4gl01 (a, na, n, mt, mg, v, iwk, wk);

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	a	{D∗ / R∗}	na×n	Input	Matrix in which observed values are stored ($x_{ij(kl)}$)
2	na	I	1	Input	Adjustable dimension of arrays a, mt and mg
3	n	I	1	Input	Number of trials n
4	mt	I*	na×n	Input	Latin square MT
5	mg	I*	na×n	Input	Latin square MG
6	v	{D∗ / R∗}	19	Output	Analysis of variance results (See Table 7−11 in note (a))
7	iwk	I*	n×n	Work	Work area
8	wk	{D∗ / R∗}	n	Work	Work area
9	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) na≥n≥3

(b) mt and mg are Latin squares.

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
1000	n=3 was specified.	The absolute value maximum that can be represented is set for the unbiased variance and variance ratio.
3000	Restriction (a) was not satisfied.	Processing is aborted.
3010	Restriction (b) was not satisfied.	

(6) **Notes**

(a) The analysis of variance results are stored as follows in the array v.

Table 7−11 Analysis of Variance Table

Factor	Variation	Degrees of freedom	Unbiased variance	Variance ratio
P (row)	v[0]	v[5]	v[10]	v[15]
Q (column)	v[1]	v[6]	v[11]	v[16]
R	v[2]	v[7]	v[12]	v[17]
A	v[3]	v[8]	v[13]	v[18]
Error	v[4]	v[9]	v[14]	

(b) The factors are allocated so that factors P and Q are in a two-dimensional arrangement, and factors R and A are associated with Latin squares MT and MG for using the various levels.

Table 7−12 Experiment Allocation for Greco-Latin Square Method

	Q_1	Q_2	Q_3	Q_4
P_1	R_2A_1	R_4A_3	R_1A_2	R_3A_4
P_2	R_4A_2	R_2A_4	R_3A_1	R_1A_3
P_3	R_1A_4	R_3A_2	R_2A_3	R_4A_1
P_4	R_3A_3	R_1A_1	R_4A_4	R_2A_2

(7) **Example**

(a) Problem

When the observed values X and two Latin squares MT and MG are given as shown below, perform an analysis of variance using the Greco-Latin square method.

$$X = \begin{bmatrix} 8.6 & 11.0 & 17.2 & 18.2 \\ 13.5 & 13.4 & 20.3 & 19.0 \\ 14.8 & 20.5 & 16.9 & 18.7 \\ 18.7 & 17.2 & 20.7 & 22.8 \end{bmatrix}$$

$$MT = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 2 & 1 & 4 & 3 \\ 3 & 4 & 1 & 2 \\ 4 & 3 & 2 & 1 \end{bmatrix}$$

$$MG = \begin{bmatrix} 1 & 3 & 4 & 2 \\ 2 & 4 & 3 & 1 \\ 3 & 1 & 2 & 4 \\ 4 & 2 & 1 & 3 \end{bmatrix}$$

(b) Input data

Observation matrix X, Latin squares MT and MG, na=100 and n=4.

(c) Main program

```
/*      C interface example for ASL_d4gl01 */

#include <stdio.h>
#include <stdlib.h>
#include <asl.h>
```

```
int main()
{
    double *a;
    int na, n;
    int *mt, *mg;
    double v[19];
    int *iwk;
    double *wk;
    int ierr;
    int i,j;
    FILE *fp;

    fp = fopen( "d4gl01.dat", "r" );
    if( fp == NULL )
    {
        printf( "file open error\n" );
        return -1;
    }

    printf( "    *** ASL_d4gl01 ***\n" );
    printf( "\n     ** Input **\n\n" );

    fscanf( fp, "%d", &na );
    fscanf( fp, "%d", &n );

    a = ( double * )malloc((size_t)( sizeof(double) * (na*n) ));
    if( a == NULL )
    {
        printf( "no enough memory for array a\n" );
        return -1;
    }

    mt = ( int * )malloc((size_t)( sizeof(int) * (na*n) ));
    if( mt == NULL )
    {
        printf( "no enough memory for array mt\n" );
        return -1;
    }

    mg = ( int * )malloc((size_t)( sizeof(int) * (na*n) ));
    if( mg == NULL )
    {
        printf( "no enough memory for array mg\n" );
        return -1;
    }

    iwk = ( int * )malloc((size_t)( sizeof(int) * (n*n) ));
    if( iwk == NULL )
    {
        printf( "no enough memory for array iwk\n" );
        return -1;
    }

    wk = ( double * )malloc((size_t)( sizeof(double) * n ));
    if( wk == NULL )
    {
        printf( "no enough memory for array wk\n" );
        return -1;
    }

    printf( "\tna = %6d\n", na );
    printf( "\tn  = %6d\n", n  );

    printf("\n\tObservations\n\n");
    for( i=0 ; i<n ; i++ )
    {
        printf( "\t" );
        for( j=0 ; j<n ; j++ )
        {
            fscanf( fp, "%lf", &a[i+na*j] );
            printf( "%8.3g ", a[i+na*j] );
        }
        printf( "\n" );
    }

    printf("\n\tLatin square mt\n\n");
    for( i=0 ; i<n ; i++ )
    {
        printf( "\t" );
        for( j=0 ; j<n ; j++ )
        {
            fscanf( fp, "%d", &mt[i+na*j] );
            printf( "%6d ", mt[i+na*j] );
        }
        printf( "\n" );
    }

    printf("\n\tLatin square mg\n\n");
    for( i=0 ; i<n ; i++ )
    {
```

```
            printf( "\t" );
            for( j=0 ; j<n ; j++ )
            {
                fscanf( fp, "%d", &mg[i+na*j] );
                printf( "%6d ", mg[i+na*j] );
            }
            printf( "\n" );
        }

        fclose( fp );

        ierr = ASL_d4gl01(a, na, n, mt, mg, v, iwk, wk);

        printf( "\n     ** Output **\n\n" );
        printf( "\tierr = %6d\n", ierr );

        printf( "\n\tAnalysis of variance table\n\n" );
        printf( "\tFactor         S.S.       D.F.   M.S.    V.R.\n" );
        printf( "\t-----------------------------------------------\n" );
        printf( "\tRow        %8.3g %8.3g %8.3g %8.3g\n",
            v[0],v[5],v[10],v[15]);
        printf( "\tColumn     %8.3g %8.3g %8.3g %8.3g\n",
            v[1],v[6],v[11],v[16]);
        printf( "\tTreatment R%8.3g %8.3g %8.3g %8.3g\n",
            v[2],v[7],v[12],v[17]);
        printf( "\tTreatment A%8.3g %8.3g %8.3g %8.3g\n",
            v[3],v[8],v[13],v[18]);
        printf( "\tError      %8.3g %8.3g %8.3g\n",
            v[4],v[9],v[14]);

        free( a );
        free( mt );
        free( mg );
        free( iwk );
        free( wk );

        return 0;
    }
```

(d) Output results

```
*** ASL_d4gl01 ***

 ** Input **

na =    100
n  =      4

Observations

     8.6       11     17.2     18.2
    13.5     13.4     20.3       19
    14.8     20.5     16.9     18.7
    18.7     17.2     20.7     22.8

Latin square mt

     1      2      3      4
     2      1      4      3
     3      4      1      2
     4      3      2      1

Latin square mg

     1      3      4      2
     2      4      3      1
     3      1      2      4
     4      2      1      3

 ** Output **

ierr =      0

Analysis of variance table

Factor         S.S.       D.F.   M.S.    V.R.
-----------------------------------------------
Row            77.6          3     25.9     5.89
Column         88.4          3     29.5      6.7
Treatment R    37.6          3     12.5     2.85
Treatment A    1.56          3    0.519    0.118
Error          13.2          3      4.4
```

7.8 BALANCED INCOMPLETE BLOCK DESIGN

7.8.1 ASL_d4bi01, ASL_r4bi01
Analysis of Variance Using Balanced Incomplete Block Design

(1) **Function**

The ASL_d4bi01 or ASL_r4bi01 performs an analysis for balanced incomplete block design data. An incomplete block design indicates the case in which one set of trials (treatments) to be compared is incomplete and is not entered in the blocks. In particular, a block design in which the number of repetitions of each trial is equal and the number of times two arbitrary trials appear in the same block is equal is called a balanced incomplete block design.

The following example shows a situation in which the number of blocks is 4, number of trials is 4, and number of trials per block is 3.

Blocks	Trials				$T_{i\cdot}$
	1	2	3	4	
A_1	x_{11}		x_{13}	x_{14}	$T_{1\cdot}$
A_2		x_{22}	x_{23}	x_{24}	$T_{2\cdot}$
A_3	x_{31}	x_{32}	x_{33}		$T_{3\cdot}$
A_4	x_{41}	x_{42}		x_{44}	$T_{4\cdot}$
$T_{\cdot j}$	$T_{\cdot 1}$	$T_{\cdot 2}$	$T_{\cdot 3}$	$T_{\cdot 4}$	T

For this example, the number of times two arbitrary trials appear in the same block at the same time is 2. Given the data $\{x_{ij}\}$, $(i = 1, \cdots, n; j = 1, \cdots, t)$, which is allocated using a balanced incomplete block design for which the number of blocks is n, number of trials is t, and number of trials per block is m, the analysis of variance results are defined as follows.

Total sum :

$$T = \sum_{i=1}^{n} \sum_{j=1}^{t} n_{ij} x_{ij}$$

Total sum for each block :

$$T_{i\cdot} = \sum_{j=1}^{t} n_{ij} x_{ij}$$

Total sum for each trial :

$$T_{\cdot j} = \sum_{i=1}^{n} n_{ij} x_{ij}$$

Total sum for each block-adjusted trial :

$$Q_{\cdot j} = m \cdot T_{\cdot j} - \sum_{i=1}^{n} (n_{ij} \cdot T_{i\cdot})$$

$(n_{ij} : (i, j)$ element of incidence matrix $N)$

Variation :

- Total variation

$$S = \sum_{i=1}^{n} \sum_{j=1}^{t} n_{ij} x_{ij}^2 - CF$$

  Here, CF is defined as follows.

$$CF = \frac{T^2}{n \cdot m} = \frac{1}{n \cdot m} \cdot \left(\sum_{i=1}^{n} \sum_{j=1}^{t} n_{ij} x_{ij} \right)^2$$

- Inter-block variation

$$S_B = \frac{1}{m} \sum_{i=1}^{n} T_{i\cdot}^2 - CF$$

- Inter-trial variation (block adjusted)

$$S_T = \frac{t-1}{n \cdot m^2 \cdot (m-1)} \sum_{j=1}^{t} Q_{\cdot j}^2$$

- Error variation

$$S_E = S - (S_B + S_T)$$

Degrees of freedom :

$$\begin{aligned} \phi &= n \cdot m - 1 \\ \phi_B &= n - 1 \\ \phi_T &= t - 1 \\ \phi_E &= n \cdot m - n - t + 1 \end{aligned}$$

Unbiased variance :

$$\begin{aligned} V_T &= \frac{S_T}{\phi_T} \\ V_B &= \frac{S_B}{\phi_B} \\ V_E &= \frac{S_E}{\phi_E} \end{aligned}$$

Variance ratio :

$$F_T = \frac{V_T}{V_B}$$

(2) **Usage**

Double precision:

ierr = ASL_d4bi01 (a, na, nb, nt, m, n, v, w1);

Single precision:

ierr = ASL_r4bi01 (a, na, nb, nt, m, n, v, w1);

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	a	{D∗ / R∗}	na×nt	Input	Matrix in which observed values are stored (x_{ij})
2	na	I	1	Input	Adjustable dimension of arrays a and n
3	nb	I	1	Input	Total number of blocks n
4	nt	I	1	Input	Number of trials t
5	m	I	1	Input	Number of trials per block m
6	n	{D∗ / R∗}	na×nt	Input	Incidence matrix (n_{ij}). When trial j is performed in the i-th block, $n_{ij} = 1$, otherwise $n_{ij} = 0$.
7	v	{D∗ / R∗}	11	Output	Analysis of variance results (For the method of storing the analysis of variance results, see Table 7−13 in note (b))
8	w1	{D∗ / R∗}	nb	Work	Work area
9	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) na ≥ nb ≥ 2

(b) nt ≥ m ≥ 2

(c) $\dfrac{\text{nb} \cdot \text{m}}{\text{nt}}$ is an integer.

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
3000	Restriction (a) or (b) was not satisfied.	Processing is aborted.
3010	Restriction (c) was not satisfied.	

(6) **Notes**

(a) The analysis of variance results are stored as follows in the array v.

Table 7−13 Analysis of Variance Table

Element	Variation	Degrees of freedom	Unbiased variance	Variance ratio
Sum of squares of observed values	v[0]	v[4]		
Inter-block	v[1]	v[5]	v[8]	v[11]
Inter-trial (block adjusted)	v[2]	v[6]	v[9]	
Error	v[3]	v[7]	v[10]	

(7) **Example**

(a) Problem

Perform an analysis of variance for the data allocated using a balanced incomplete block design given by the matrix A shown below. The asterisks ($*$) represent portions for which no corresponding observed values exist.

$$A = \begin{bmatrix} 2 & * & 4 & 0 \\ * & 32 & 13 & 23 \\ 20 & 14 & 31 & * \\ 7 & 3 & * & 11 \end{bmatrix}$$

The incidence matrix N is given as follows.

$$N = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix}$$

(b) Input dataData allocated by a balanced incomplete block design A, incidence matrix N, na=4, nb=4, nt=4 and m=3.

(c) Main program

```
/*      C interface example for ASL_d4bi01 */

#include <stdio.h>
#include <stdlib.h>
#include <asl.h>

int main()
{
    double *a;
    int na;
    int nb;
    int nt;
    int m;
    int *n;
    double v[12];
    double *w1;
    int ierr;
    int i,j;
    FILE *fp;

    fp = fopen( "d4bi01.dat", "r" );
    if( fp == NULL )
    {
        printf( "file open error\n" );
        return -1;
    }

    printf( "    *** ASL_d4bi01 ***\n" );
    printf( "\n     ** Input **\n\n" );

    fscanf( fp, "%d", &na );
    fscanf( fp, "%d", &nb );
    fscanf( fp, "%d", &nt );
    fscanf( fp, "%d", &m );

    a = ( double * )malloc((size_t)( sizeof(double) * (na*nt) ));
    if( a == NULL )
    {
        printf( "no enough memory for array a\n" );
        return -1;
    }

    n = ( int * )malloc((size_t)( sizeof(int) * (na*nt) ));
    if( n == NULL )
    {
        printf( "no enough memory for array n\n" );
        return -1;
    }

    w1 = ( double * )malloc((size_t)( sizeof(double) * nb ));
    if( w1 == NULL )
    {
        printf( "no enough memory for array w1\n" );
        return -1;
```

```
    }

    printf( "\tna = %6d nb = %6d\n", na, nb );
    printf( "\tnt = %6d m  = %6d\n", nt, m );

    printf( "\n\t Matrix a\n\n");
    for( i=0 ; i<nb ; i++ )
    {
        for( j=0 ; j<nt ; j++ )
        {
            fscanf( fp, "%lf", &a[i+na*j] );
            printf( "%8.3g", a[i+na*j] );
        }
        printf( "\n" );
    }

    printf( "\n\t Matrix n\n\n");
    for( i=0 ; i<nb ; i++ )
    {
        for( j=0 ; j<nt ; j++ )
        {
            fscanf( fp, "%d", &n[i+na*j] );
            printf( "%6d", n[i+na*j] );
        }
        printf( "\n" );
    }

    fclose( fp );

    ierr = ASL_d4bi01(a, na, nb, nt, m, n, v, w1);

    printf( "\n     ** Output **\n\n" );
    printf( "\tierr = %6d\n", ierr );

    printf( "\n\tAnalysis of variance table\n\n" );
    printf( "\t Factor      S.S.     D.F.     M.S.    V.R.\n" );
    printf( "\t-------------------------------------------\n" );
    printf( "\t Total  %8.3g %8.3g\n", v[0],v[4] );
    printf( "\t  SB    %8.3g %8.3g %8.3g %8.3g\n", v[1],v[5],v[8],v[11]);
    printf( "\t  ST    %8.3g %8.3g %8.3g\n", v[2],v[6],v[9]);
    printf( "\t Error  %8.3g %8.3g %8.3g\n",v[3],v[7],v[10]);

    free( a );
    free( n );
    free( w1 );

    return 0;
}
```

(d) Output results

```
*** ASL_d4bi01 ***

 ** Input **

na =      4 nb =      4
nt =      4 m  =      3

 Matrix a

   2       0       4       0
   0      32      13      23
  20      14      31       0
   7       3       0      11

 Matrix n

 1     0     1     1
 0     1     1     1
 1     1     1     0
 1     1     0     1

 ** Output **

ierr =      0

Analysis of variance table

 Factor      S.S.     D.F.     M.S.    V.R.
-------------------------------------------
 Total  1.34e+03       11
  SB         975        3      325  0.00632
  ST        6.17        3     2.06
 Error       363        5     72.6
```

Chapter 8

NONPARAMETRIC TESTS

8.1 INTRODUCTION

In traditional statistical tests, a normal distribution often is assumed as the population distribution. It is also a prerequisite that the observed values are continuous quantities. A test of this kind for which the population distribution is fixed and the observed values are assumed to be continuous quantities is called a **parametric test**. In contrast, a test for which the population distribution is not fixed or for which the data subject to the test is represented by sequences or frequencies, not by observed values that are complete quantities, is called a **nonparametric test**.
This library provides the following functions for performing nonparametric tests.

- Test of Goodness of Fit
- χ^2 Test (2×2 Contingency Table)
- χ^2 Test ($m \times n$ Contingency Table)
- Median Test
- Sign Test
- Wilcoxon Test
- Mann-Whitney's U Test
- Spearman's Rank Correlation Test

8.1.1 Explanation

(1) **Test of Goodness of Fit**

Given observed frequencies, this test tests whether or not their distribution is equal to a theoretical distribution postulated by the researcher. The goodness-of-fit test is performed by obtaining the χ^2 value for the test for the theoretical probabilities $p_i,\ (i = 1, \cdots, n)$ and observed frequencies $f_i,\ (i = 1, \cdots, n)$ of n classes as defined below:

$$\chi^2 = \sum_{i=1}^{n} \frac{(f_i - e_i)^2}{e_i}$$

and comparing this value with the critical value of a χ^2 distribution with $n - 1$ degrees of freedom.

(2) χ^2 **Test (**2×2 **Contingency Table)**

This is a test for testing the independence of two factors by using a 2×2 contingency table.

	A	$\bar{A}$	Sum
B	a_{11}	a_{12}	$a_{11} + a_{12}$
$\bar{B}$	a_{21}	a_{22}	$a_{21} + a_{22}$
Sum	$a_{11} + a_{21}$	$a_{12} + a_{22}$	n

In the 2×2 contingency table shown above, A means that the subject has the characteristic being observed by factor A, and $\bar{A}$ means that the subject does not have the characteristic. B and $\bar{B}$ have similar meanings for factor B. Also, a_{ij} represents the actual measured frequency of the corresponding characteristics. The χ^2 value, which is the test quantity, for the above contingency table is given as follows.

$$\chi^2 = \frac{n\left(|a_{11}a_{22} - a_{12}a_{21}| - \dfrac{n}{2}\right)^2}{(a_{11} + a_{12})(a_{21} + a_{22})(a_{11} + a_{21})(a_{12} + a_{22})}$$

Here, $n = a_{11} + a_{12} + a_{21} + a_{22}$. This equation contains Yates' correction for continuity term $-\frac{n}{2}$. The χ^2 test is performed by comparing this χ^2 value with the critical value of a χ^2 distribution with 1 degree of freedom.

(3) χ^2 **Test (**$m \times n$ **Contingency Table)**

This is a test for testing the independence of two factors by using an $m \times n$ contingency table.

	B_1	B_2	$\cdots$	B_j	$\cdots$	B_n	Sum
A_1	a_{11}	a_{12}		$\vdots$		a_{1n}	$a_{1\cdot}$
A_2	a_{21}	a_{22}		$\vdots$		a_{2n}	$a_{2\cdot}$
$\vdots$				$\vdots$			$\vdots$
A_i	$\cdots$	$\cdots$	$\cdots$	a_{ij}	$\cdots$	$\cdots$	$a_{i\cdot}$
$\vdots$				$\vdots$			$\vdots$
A_m	a_{m1}	a_{m2}		$\vdots$		a_{mn}	$a_{m\cdot}$
Sum	$a_{\cdot 1}$	$a_{\cdot 2}$	$\cdots$	$a_{\cdot j}$	$\cdots$	$a_{\cdot n}$	S

In the $m \times n$ contingency table shown above, A_i means that the subject has the ith level of the characteristic being observed by factor A, and each level is assumed to be independent. The meanings are similar for B_j and factor B. Also, a_{ij} represents the actual measured frequency of the subject having the characteristics

for both the ith level of factor A and the jth level of factor B. The χ^2 value for the above contingency table is given as follows.

$$\chi^2 = \sum_{i=1}^{m}\sum_{j=1}^{n}\frac{(a_{ij}-e_{ij})^2}{e_{ij}}$$

The χ^2 test is performed by obtaining this χ^2 value and comparing it with the critical value of a χ^2 distribution with $(m-1)(n-1)$ degrees of freedom. Here, e_{ij} is defined according to the following values.
Row sum :

$$a_{i\cdot} = \sum_{j=1}^{n} a_{ij}$$

Column sum :

$$a_{\cdot j} = \sum_{i=1}^{m} a_{ij}$$

Total sum :

$$S = \sum_{i=1}^{m}\sum_{j=1}^{n} a_{ij}$$

Expected value :

$$e_{ij} = \frac{a_{i\cdot}\cdot a_{\cdot j}}{S}$$

(4) **Median Test**
The median test tests the hypothesis that the median values of two groups are equal. The test is performed by obtaining the χ^2 value and comparing it with the critical value of a χ^2 distribution with one degree of freedom.

(5) **Sign Test**
When the observed values of two samples are individually associated and their relative sizes can be judged (including determining that they are equal), the sign test tests the hypothesis that it is equally probable that the judgements "it is larger" $(+)$ and "it is smaller" $(-)$ will appear. The critical value of a standard normal distribution is used for the test.

(6) **Wilcoxon Test**
The Wilcoxon test is a more precise test than the sign test. The critical value of a standard normal distribution is used for the test.

(7) **Mann-Whitney's U Test**
When two independent groups of samples are given, Mann-Whitney's U Test tests whether or not the distributions of the populations to which those samples belong are equal. The critical value of a standard normal distribution is used for the test.

(8) **Spearman's Rank Correlation Test**
When n pairs of observed values (x_i, y_i), $(i = 1, \cdots, n)$ are given, if x_i and y_i are each independently

ranked and the rankings are represented by $R(x_i)$ and $R(y_i)$, the Spearman's rank correlation coefficient r_s is defined as follows.

$$r_s = 1 - \frac{6\sum_{i=1}^{n}(R(x_i) - R(y_i))^2}{n^3 - n}$$

Spearman's rank correlation coefficient is an index representing the strength of the association of $R(x_i)$ and $R(y_i)$. This index can be used to test the hypothesis that there is no correlation between x_i and y_i.

8.1.2 Reference Bibliography

(1) Gibbons, J. D. , "Nonparametric Statistical Inference", McGraw-Hill (1971)

(2) Lehmann, E. L. , "Nonparametrics: Statistical methods based on ranks", Holden-Day, San Francisco (1975)

8.2 TESTS USING χ^2 DISTRIBUTION

8.2.1 ASL_d5chef, ASL_r5chef
Test of Goodness of Fit

(1) **Function**

The ASL_d5chef or ASL_r5chef tests the goodness of fit to the expected frequencies (theoretical frequencies) of given observed frequencies. The values for the theoretical probabilities $p_i,\ (i = 1, \cdots, n)$ and observed frequencies $f_i,\ (i = 1, \cdots, n)$ of n classes are defined by the equations below.

Total of frequencies of all classes :

$$S = \sum_{i=1}^{n} f_i$$

Expected frequency (theoretical frequency) of ith class :

$$e_i = S \cdot p_i$$

χ^2 value for test :

$$\chi^2 = \sum_{i=1}^{n} \frac{(f_i - e_i)^2}{e_i}$$

Degrees of freedom :

$$\phi = n - 1$$

(2) **Usage**

Double precision:

```
ierr = ASL_d5chef  (p, n, f, &idf, &chi);
```

Single precision:

```
ierr = ASL_r5chef  (p, n, f, &idf, &chi);
```

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	p	{D∗ / R∗}	n	Input	Theoretical probabilities $\{p_i\}$
2	n	I	1	Input	Number of classes of expected frequencies n
3	f	{D∗ / R∗}	n	Input	Observed frequencies $\{f_i\}$
4	idf	I*	1	Output	Degrees of freedom ϕ
5	chi	{D∗ / R∗}	1	Output	χ^2 value
6	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) n≥2

(b) 0<p[i]<1 (i=0, ⋯, n-1)

(c) $\sum_{i=0}^{n-1} p[i] \leq 1$

(d) f[i]≥ 0 (i=0, ⋯, n-1)

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
3000	Restriction (a) was not satisfied.	Processing is aborted.
3010	Restriction (b) was not satisfied.	
3020	Restriction (c) was not satisfied.	
3030	Restriction (d) was not satisfied.	

(6) **Notes**

(a) For the given theoretical probabilities $p[i-1]$, $(i = 1, \cdots, n)$, the error is reduced by replacing $p[n-1]$ by the following:

$$p[n-1] = 1 - \sum_{i=1}^{n-1} p[i-1]$$

(7) **Example**

(a) Problem

When the number of classes is 7 and the theoretical probability p_i and observed frequency f_i of each class are given as shown below, perform a test of the goodness of fit to the expected frequencies (theoretical frequencies) of the observed frequencies.

Class	p_i	f_i
1	0.1	10
2	0.2	20
3	0.3	30
4	0.1	10
5	0.1	10
6	0.12	12
7	0.05	5

(b) Input data

Theoretical probabilities p_i, observed frequencies f_i and n = 7.

(c) Main program

```
/*      C interface example for ASL_d5chef */

#include <stdio.h>
#include <stdlib.h>
#include <asl.h>

int main()
{
    double *p;
    int n;
    double *f;
    int idf;
    double chi;
    int ierr;
    int i;
    FILE *fp;

    fp = fopen( "d5chef.dat", "r" );
    if( fp == NULL )
    {
        printf( "file open error\n" );
        return -1;
    }

    printf( "    *** ASL_d5chef ***\n" );
    printf( "\n     ** Input **\n\n" );

    fscanf( fp, "%d", &n );

    p = ( double * )malloc((size_t)( sizeof(double) * n ));
    if( p == NULL )
    {
        printf( "no enough memory for array p\n" );
        return -1;
    }

    f = ( double * )malloc((size_t)( sizeof(double) * n ));
    if( f == NULL )
    {
        printf( "no enough memory for array f\n" );
        return -1;
    }

    printf( "\tn = %6d\n\n", n );

    printf( "\tProbability  Frequency\n" );
    printf( "\t----------------------\n" );
    for( i=0 ; i<n ; i++ )
    {
        fscanf( fp, "%lf %lf", &p[i],&f[i] );
        printf( "\t%8.3g     %8.3g\n", p[i],f[i] );
    }

    fclose( fp );

    ierr = ASL_d5chef(p, n, f, &idf, &chi);
```

```
    printf( "\n     ** Output **\n\n" );
    printf( "\tierr = %6d\n", ierr );

    printf( "\n\tDegree of freedom  = %6d\n", idf );
    printf( "\n\tChi square value   = %6.3g\n", chi );

    free( p );
    free( f );

    return 0;
}
```

(d) Output results

```
*** ASL_d5chef ***

 ** Input **

n =      7

Probability  Frequency
----------------------
     0.1            10
     0.2            20
     0.3            30
     0.1            10
     0.1            10
    0.12            12
    0.05             5

 ** Output **

ierr =      0

Degree of freedom  =      6

Chi square value   =   1.07
```

8.2.2 ASL_d5chtt, ASL_r5chtt
χ^2 Test (2×2 Contingency Table)

(1) **Function**

The ASL_d5chtt or ASL_r5chtt obtains the χ^2 value, which is the test quantity including the Yates' correction for continuity term, by using a 2×2 contingency table.

	A	$\bar{A}$	Sum
B	a_{11}	a_{12}	$a_{11}+a_{12}$
$\bar{B}$	a_{21}	a_{22}	$a_{21}+a_{22}$
Sum	$a_{11}+a_{21}$	$a_{12}+a_{22}$	n

In the 2×2 contingency table shown above, A means that the subject has the characteristic being observed by factor A, and $\bar{A}$ means that the subject does not have the characteristic. B and $\bar{B}$ have similar meanings for factor B. Also, a_{ij} represents the actual measured frequency of the corresponding characteristics. The χ^2 value, which is the test quantity, for the above contingency table is given as follows.

$$\chi^2 = \frac{n\left(|a_{11}a_{22} - a_{12}a_{21}| - \dfrac{n}{2}\right)^2}{(a_{11}+a_{12})(a_{21}+a_{22})(a_{11}+a_{21})(a_{12}+a_{22})}$$

Here, $n = a_{11} + a_{12} + a_{21} + a_{22}$.

The χ^2 test tests the independence of two factors by comparing this χ^2 value with the critical value of a χ^2 distribution with 1 degree of freedom.

(2) **Usage**

Double precision:

```
ierr = ASL_d5chtt  (f, &chi);
```

Single precision:

```
ierr = ASL_r5chtt  (f, &chi);
```

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	f	{ D∗ / R∗ }	2×2	Input	Observed frequencies constituting the contingency table a_{ij}
2	chi	{ D∗ / R∗ }	1	Output	χ^2 value
3	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) f[i]≥0 (i=0, ···, 3)

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
1000	All observed frequencies were zero.	The absolute value maximum that can be represented is set for the χ^2 value.
3000	Restriction (a) was not satisfied.	Processing is aborted.

(6) **Notes**

(a) The observed frequencies of the contingency table are stored in array f as the following real matrix (two-dimensional array type) (See Appendix A).

$$\begin{bmatrix} a_{11} & a_{21} \\ a_{21} & a_{22} \end{bmatrix}$$

(7) **Example**

(a) Problem

Obtain the χ^2 value, which is the test quantity including the Yates' correction for continuity term, by using the following 2×2 contingency table.

	A	$\bar{A}$	Sum
B	6	0	6
$\bar{B}$	1	3	4
Sum	7	3	10

(b) Input data

f[0] = 6.0, f[1] = 1.0, f[2] = 0.0 and f[3] = 3.0.

(c) Main program

```
/*      C interface example for ASL_d5chtt */

#include <stdio.h>
#include <stdlib.h>
#include <asl.h>

int main()
{
    double f[2*2];
    double chi;
    int ierr;
    int i,j;
    FILE *fp;

    fp = fopen( "d5chtt.dat", "r" );
    if( fp == NULL )
    {
        printf( "file open error\n" );
        return -1;
    }

    printf( "    *** ASL_d5chtt ***\n" );
    printf( "\n     ** Input **\n\n" );

    printf("\tTwo-by-two contingency table\n\n");
    for( i=0 ; i<2 ; i++ )
    {
        printf( "\t" );
        for( j=0 ; j<2 ; j++ )
        {
            fscanf( fp, "%lf", &f[i+2*j] );
            printf( "%8.3g ", f[i+2*j] );
        }
        printf( "\n" );
    }
```

```
    fclose( fp );

    ierr = ASL_d5chtt(f, &chi);

    printf( "\n     ** Output **\n\n" );
    printf( "\tierr = %6d\n", ierr );

    printf( "\n\tChi square value = %8.3g\n", chi );

    return 0;
}
```

(d) Output results

```
*** ASL_d5chtt ***

 ** Input **

Two-by-two contingency table

       6        0
       1        3

 ** Output **

ierr =      0

Chi square value =     3.35
```

8.2.3 ASL_d5chmn, ASL_r5chmn
χ^2 Test ($m \times n$ Contingency Table)

(1) **Function**

The ASL_d5chmn or ASL_r5chmn obtains the χ^2 value, which is the test quantity, by using an $m \times n$ contingency table.

	B_1	B_2	$\cdots$	B_j	$\cdots$	B_n	Sum
A_1	a_{11}	a_{12}		$\vdots$		a_{1n}	$a_{1\cdot}$
A_2	a_{21}	a_{22}		$\vdots$		a_{2n}	$a_{2\cdot}$
$\vdots$				$\vdots$			$\vdots$
A_i	$\cdots$	$\cdots$	$\cdots$	a_{ij}	$\cdots$	$\cdots$	$a_{i\cdot}$
$\vdots$				$\vdots$			$\vdots$
A_m	a_{m1}	a_{m2}		$\vdots$		a_{mn}	$a_{m\cdot}$
Sum	$a_{\cdot 1}$	$a_{\cdot 2}$	$\cdots$	$a_{\cdot j}$	$\cdots$	$a_{\cdot n}$	S

In the $m \times n$ contingency table shown above, A_i means that the subject has the ith level of the characteristic being observed by factor A, and each level is assumed to be independent. The meanings are similar for B_j and B. Also, a_{ij} represents the actual measured frequency of the subject having the characteristics for both the ith level of factor A and the jth level of factor B. The χ^2 value, which is the test quantity, for the above contingency table is given as follows.

$$\chi^2 = \sum_{i=1}^{m} \sum_{j=1}^{n} \frac{(a_{ij} - e_{ij})^2}{e_{ij}}$$

Here, the expected values e_{ij} are defined as follows.
Expected value :

$$e_{ij} = \frac{a_{i\cdot} \cdot a_{\cdot j}}{S}$$

Row sum :

$$a_{i\cdot} = \sum_{j=1}^{n} a_{ij}$$

Column sum :

$$a_{\cdot j} = \sum_{i=1}^{m} a_{ij}$$

Total sum :

$$S = \sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij}$$

The χ^2 test tests the independence of two factors by comparing this χ^2 value with the critical value of a χ^2 distribution with $(m-1)(n-1)$ degrees of freedom.

(2) **Usage**

Double precision:

```
ierr = ASL_d5chmn  (a, na, m, n, &idf, &chi, wk);
```

Single precision:

```
ierr = ASL_r5chmn  (a, na, m, n, &idf, &chi, wk);
```

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	a	{D∗ / R∗}	na×n	Input	Observed frequencies constituting the contingency table (a_{ij})
2	na	I	1	Input	Adjustable dimension of array a
3	m	I	1	Input	Number of rows of contingency table m
4	n	I	1	Input	Number of columns of contingency table n
5	idf	I*	1	Output	Degrees of freedom ϕ
6	chi	{D∗ / R∗}	1	Output	χ^2 value
7	wk	{D∗ / R∗}	max(m, n)	Work	Work area
8	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) na≥m≥2

(b) n≥2

(c) a[(i-1)+(j-1)×na]≥0 (i=1, ⋯, m ; j=1, ⋯, n)

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
1000	The expected value in one or more cells was less than or equal to 1.0.	chi and idf are calculated.
3000	Restriction (a), (b) was not satisfied.	Processing is aborted.
3010	Restriction (c) was not satisfied.	
4000	All observation frequencies were 0.0 or the expected value was less than or equal to 0.0.	

(6) **Notes**

(a) The observed frequencies of the contingency table are stored in array f as an $m \times n$ real matrix (two-dimensional array type) defined as $A = (a_{i,j})$ (See Appendix A).

(7) **Example**

(a) Problem
Obtain the χ^2 value by using the following 3×4 matrix, which corresponds to a 3×4 contingency table of observed frequencies.

$$\begin{bmatrix} 34 & 50 & 24 & 12 \\ 22 & 65 & 115 & 24 \\ 12 & 41 & 68 & 20 \end{bmatrix}$$

(b) Input data

Array corresponding to observed frequencies a, na = 5, m = 3 and n = 4.

(c) Main program

```
/*      C interface example for ASL_d5chmn */

#include <stdio.h>
#include <stdlib.h>
#include <asl.h>

int main()
{
    double *a;
    int na;
    int m;
    int n;
    int idf;
    double chi;
    double *wk;
    int ierr;
    int i,j;
    FILE *fp;

    fp = fopen( "d5chmn.dat", "r" );
    if( fp == NULL )
    {
        printf( "file open error\n" );
        return -1;
    }

    printf( "    *** ASL_d5chmn ***\n" );
    printf( "\n     ** Input **\n\n" );

    fscanf( fp, "%d", &na );
    fscanf( fp, "%d", &m );
    fscanf( fp, "%d", &n );

    a = ( double * )malloc((size_t)( sizeof(double) * (na*n) ));
    if( a == NULL )
    {
        printf( "no enough memory for array a\n" );
        return -1;
    }

    wk = ( double * )malloc((size_t)( sizeof(double) * n ));
    if( wk == NULL )
    {
        printf( "no enough memory for array wk\n" );
        return -1;
    }

    printf( "\tna=%2d,  m=%2d,  n=%2d\n", na, m, n );

    printf("\n\tContingency table\n\n");
    for( i=0 ; i<m ; i++ )
    {
        printf( "\t" );
        for( j=0 ; j<n ; j++ )
        {
            fscanf( fp, "%lf", &a[i+na*j] );
            printf( "%8.3g ", a[i+na*j] );
        }
        printf( "\n" );
```

```
        }

        fclose( fp );

        ierr = ASL_d5chmn(a, na, m, n, &idf, &chi, wk);

        printf( "\n     ** Output **\n\n" );
        printf( "\tierr = %6d\n", ierr );
        printf( "\n\tDegree of freedom = %6d\n", idf );
        printf( "\n\tChi square value  = %7.3g\n", chi );

        free( a );
        free( wk );

        return 0;
    }
```

(d) Output results

```
*** ASL_d5chmn ***

 ** Input **

na= 5,  m= 3,  n= 4

Contingency table

      34       50       24       12
      22       65      115       24
      12       41       68       20

 ** Output **

ierr =      0

Degree of freedom =      6

Chi square value  =    48.6
```

8.2.4 ASL_d5chmd, ASL_r5chmd Median Test

(1) **Function**

The ASL_d5chmd or ASL_r5chmd obtains the χ^2 value, which is the test quantity, by using the median test for two independent samples.

The value for the observation values $x_i,\ (i = 1, 2, \cdots, n)$ and $y_j,\ (j = 1, 2, \cdots, m)$ of two samples is defined by the following equation.

χ^2 value :

$$\chi^2 = \frac{(n+m)\left(\mid a \cdot d - b \cdot c \mid -\dfrac{n+m}{2}\right)^2}{(a+b)(c+d)(a+c)(b+d)}$$

Here, a, b, c, and d indicate the following values.

a: Among x_i, number of observed values that are larger than the median value of the $(n+m)$ observed values

b: Among x_i, number of observed values that are smaller than the median value of the $(n+m)$ observed values

c: Among y_j, number of observed values that are larger than the median value of the $(n+m)$ observed values

d: Among y_j, number of observed values that are smaller than the median value of the $(n+m)$ observed values

However, for observed values that are equal to the median value of the $(n+m)$ observed values, frequencies of 0.5 each are added to both the number that are larger and the number that are smaller.

The median test tests the hypothesis that the median values of two groups are equal by comparing this χ^2 value with the critical value of a χ^2 distribution with one degree of freedom.

(2) **Usage**

Double precision:

```
ierr = ASL_d5chmd  (a, n, b, m, &chi, wk);
```

Single precision:

```
ierr = ASL_r5chmd  (a, n, b, m, &chi, wk);
```

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	a	{D∗ / R∗}	n	Input	Observed values of sample A $\{x_i\}$
2	n	I	1	Input	Number of observed values of sample A n
3	b	{D∗ / R∗}	m	Input	Observed values of sample B $\{y_i\}$
4	m	I	1	Input	Number of observed values of sample B m
5	chi	{D∗ / R∗}	1	Output	χ^2 value
6	wk	{D∗ / R∗}	n+m	Work	Work area
7	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) n≥2

(b) m≥2

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
1000	Sample A and sample B are mutually prime.	0.0 is set for chi.
3000	Restriction (a), (b) was not satisfied.	Processing is aborted.

(6) **Notes**

None

(7) **Example**

(a) Problem

Given the following observed values for two independent samples,

$$\{x_i\} = \{160, 160, 140, 190\}$$

and

$$\{y_i\} = \{117, 145, 147, 120, 150, 120\}$$

obtain the χ^2 value by using the median test.

(b) Input data

Observed values $\{x_i\}$, n = 4, observed values $\{y_i\}$ and m = 6.

(c) Main program

```
/*      C interface example for ASL_d5chmd */

#include <stdio.h>
#include <stdlib.h>
#include <asl.h>

int main()
{
    double *a;
    int n;
    double *b;
    int m;
    double chi;
    double *wk;
    int ierr;
    int i;
    FILE *fp;

    fp = fopen( "d5chmd.dat", "r" );
    if( fp == NULL )
    {
        printf( "file open error\n" );
        return -1;
    }

    printf( "    *** ASL_d5chmd ***\n" );
    printf( "\n     ** Input **\n\n" );

    fscanf( fp, "%d", &n );
    fscanf( fp, "%d", &m );

    a = ( double * )malloc((size_t)( sizeof(double) * n ));
    if( a == NULL )
    {
        printf( "no enough memory for array a\n" );
        return -1;
    }

    b = ( double * )malloc((size_t)( sizeof(double) * m ));
    if( b == NULL )
    {
        printf( "no enough memory for array b\n" );
        return -1;
    }

    wk = ( double * )malloc((size_t)( sizeof(double) * (n+m) ));
    if( wk == NULL )
    {
        printf( "no enough memory for array wk\n" );
        return -1;
    }

    printf( "\tn=%2d,  m=%2d\n", n, m );

    printf("\n\tObservations A\n\n");
    for( i=0 ; i<n ; i++ )
    {
        fscanf( fp, "%lf", &a[i] );
        printf( " \t%8.3g\n", a[i] );
    }

    printf("\n\tObservations B\n\n");
    for( i=0 ; i<m ; i++ )
    {
        fscanf( fp, "%lf", &b[i] );
        printf( " \t%8.3g\n", b[i] );
    }

    fclose( fp );

    ierr = ASL_d5chmd(a, n, b, m, &chi, wk);

    printf( "\n     ** Output **\n\n" );
    printf( "\tierr = %6d\n", ierr );
    printf( "\n\tChi square value = %8.3g\n", chi );

    free( a );
    free( b );
    free( wk );

    return 0;
}
```

(d) Output results

```
*** ASL_d5chmd ***

 ** Input **

n= 4,  m= 6

Observations A

      16
     160
     140
     190

Observations B

     117
     145
     147
     120
     150
     120

 ** Output **

ierr =      0

Chi square value =    0.417
```

8.3 TESTS USING OTHER DISTRIBUTION

8.3.1 ASL_d5tesg, ASL_r5tesg
Sign Test

(1) **Function**

The ASL_d5tesg or ASL_r5tesg performs a sign test when the observed values of two samples are individually associated.

When (X, Y) is assumed to be a pair of random variables and n pairs of observed values (x_i, y_i), $(i = 1, 2, \cdots, n)$ are given as the actual values for them, this function tests the null hypothesis H_0 :"$P_r(X > Y) = P_r(X < Y) = 0.5$" against the alternative hypothesis H_1: "$P_r(X > Y) > 0.5$ (or < 0.5)."

The value for the n pairs of observed values (x_i, y_i), $(i = 1, 2, \cdots, n)$ is defined by the equations shown below.

The following kinds of values a, b, x, and m are obtained from the observed values.

a : Number of pairs of observed values for which $x_i > y_i$.

b : Number of pairs of observed values for which $x_i < y_i$.

$x = \min(a, b)$

$m = a + b$

Probability :

- For $m \leq 25$

$$P = \frac{1}{2^m} \sum_{i=0}^{x} \begin{pmatrix} m \\ i \end{pmatrix}$$

- For $m > 25$

$$P = \int_{-\infty}^{Z} \frac{1}{\sqrt{2 \cdot \pi}} e^{-\frac{x^2}{2}} dx$$

Here, Z is defined as follows.

$$Z = \frac{x + 0.5 - \dfrac{m}{2}}{\dfrac{1}{2}\sqrt{m}}$$

(Z obeys a standard normal distribution $N(0, 1)$.)

(2) **Usage**

Double precision:

```
ierr = ASL_d5tesg  (a, n, b, &izr, &isn, &p);
```

Single precision:

```
ierr = ASL_r5tesg  (a, n, b, &izr, &isn, &p);
```

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	a	{D∗ / R∗}	n	Input	Observed values of sample A $\{x_i\}$
2	n	I	1	Input	Number of observed values in each of the two samples n
3	b	{D∗ / R∗}	n	Input	Observed values of sample B $\{y_i\}$
4	izr	I*	1	Output	Number of times the difference of the corresponding observed values of samples A and B is not zero m
5	isn	I*	1	Output	Smaller number of appearances of the signs of the difference of the corresponding observed values (number of appearances of + or −) x
6	p	{D∗ / R∗}	1	Output	Probability (one-tailed test) P
7	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) n≥1

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
1000	Samples A and B are equal (izr=0).	isn=0 and p=1 are set.
3000	Restriction (a) was not satisfied.	Processing is aborted.

(6) **Notes**

(a) The number of observed values in the two samples must be equal.

(b) When this function is used, the test is a one-tailed test. However, when this procedure is used for a two-tailed test, if the significance level is α, the null hypothesis is rejected when the following relationship holds.

$$P \leq \frac{\alpha}{2}$$

(7) **Example**

(a) Problem

Perform a sign test for the following set of observed values of two samples.

$$\begin{aligned}\{(x_i, y_i)\} &= \{(4,2),(4,3),(5,3),(5,3),(3,3),(2,3),(5,3),(3,3),(1,2),\\ &\quad (5,3),(5,2),(5,2),(4,5),(5,2),(5,5),(5,3),(5,1)\}\end{aligned}$$

(b) Input data

Set of observed values $\{(x_i, y_i)\}$ and n=17.

(c) Main program

```
/*      C interface example for ASL_d5tesg */

#include <stdio.h>
#include <stdlib.h>
#include <asl.h>

int main()
{
    double *a;
    int n;
    double *b;
    int izr;
    int isn;
    double p;
    int ierr;
    int i;
    FILE *fp;

    fp = fopen( "d5tesg.dat", "r" );
    if( fp == NULL )
    {
        printf( "file open error\n" );
        return -1;
    }

    printf( "    *** ASL_d5tesg ***\n" );
    printf( "\n     ** Input **\n\n" );

    fscanf( fp, "%d", &n );

    a = ( double * )malloc((size_t)( sizeof(double) * n ));
    if( a == NULL )
    {
        printf( "no enough memory for array a\n" );
        return -1;
    }

    b = ( double * )malloc((size_t)( sizeof(double) * n ));
    if( b == NULL )
    {
        printf( "no enough memory for array b\n" );
        return -1;
    }

    printf( "\tn = %6d\n", n );

    printf("\n\tObservations\n\n");
    printf("\t    No.       A        B\n");
    printf("\t------------------------------\n");
    for( i=0 ; i<n ; i++ )
    {
        fscanf( fp, "%lf %lf", &a[i], &b[i] );
        printf( "\t%6d %8.3g %8.3g\n", i+1, a[i], b[i] );
    }

    fclose( fp );

    ierr = ASL_d5tesg(a, n, b, &izr, &isn, &p);

    printf( "\n     ** Output **\n\n" );
    printf( "\tierr = %6d\n", ierr );
    printf( "\n\tNumber of pairs = %6d\n", izr );
    printf( "\n\tNumber of sigs  = %6d\n", isn );
    printf( "\n\tProbability = %8.3g significant at 0.05 level\n", p );

    free( a );
    free( b );

    return 0;
}
```

(d) Output results

```
*** ASL_d5tesg ***

 ** Input **

n =     17

Observations

    No.        A         B
-----------------------------
     1         4         2
     2         4         3
     3         5         3
     4         5         3
     5         3         3
     6         2         3
     7         5         3
     8         3         3
     9         1         2
    10         5         3
    11         5         2
    12         5         2
    13         4         5
    14         5         2
    15         5         5
    16         5         3
    17         5         1

 ** Output **

ierr =      0

Number of pairs =     14

Number of sigs  =      3

Probability =   0.0287 significant at 0.05 level
```

8.3.2 ASL_d5tewl, ASL_r5tewl Wilcoxon Test

(1) **Function**

The ASL_d5tewl or ASL_r5tewl performs a Wilcoxon test when the observed values of two samples are individually associated.

When (X, Y) is assumed to be a pair of random variables and n pairs of observed values (x_i, y_i), $(i = 1, 2, \cdots, n)$ are given as the actual values for them, this function tests the null hypothesis H_0: "$P_r(X > Y) = P_r(X < Y) = 0.5$" against the alternative hypothesis H_1: "$P_r(X > Y) > 0.5$ (or < 0.5)."

The value for the n pairs of observed values (x_i, y_i), $(i = 1, 2, \cdots, n)$ is defined by the equations shown below.

Difference of corresponding observed value of two samples :

$$d_i = x_i - y_i, \quad (i = 1, \cdots, n)$$

The following kinds of values m, R_i, T_P, and T_N are obtained from the observed values.

m : Number of d_i that are not zero.

R_i : For nonzero d_i, ranks assigned to their absolute values. The mean rank is assigned for equal ranks.

T_P : Sum of ranks assigned to positive d_i

T_N : Sum of ranks assigned to negative d_i

Test statistic :

$$T = \min(T_P, T_N)$$

Expected value of T :

$$E[T] = \frac{m(m+1)}{4}$$

Variance of T:

$$V[T] = \frac{m(m+1)(2 \cdot m + 1)}{24}$$

Probability :

$$P = \int_{-\infty}^{Z} \frac{1}{\sqrt{2 \cdot \pi}} e^{-\frac{x^2}{2}} dx$$

Here, Z is defined as follows.

$$Z = \frac{T - E[T]}{\sqrt{V[T]}}$$

(The Z for which T has been normalized obeys a standard normal distribution $N(0, 1)$.)

(2) **Usage**

Double precision:

ierr = ASL_d5tewl (a, n, b, &izr, &t, &z, &p, iwk, wk);

Single precision:

ierr = ASL_r5tewl (a, n, b, &izr, &t, &z, &p, iwk, wk);

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	a	{D∗ / R∗}	n	Input	Observed values of sample A $\{x_i\}$
2	n	I	1	Input	Number of observed values in each of the two samples n
3	b	{D∗ / R∗}	n	Input	Observed values of sample B $\{y_i\}$
4	izr	I*	1	Output	Number of times the difference of the corresponding observed values of samples A and B is not zero m
5	t	{D∗ / R∗}	1	Output	Test statistic T
6	z	{D∗ / R∗}	1	Output	Value for computing the significance of t according to a normal distribution (value for which t has been normalized) Z
7	p	{D∗ / R∗}	1	Output	Probability (one-tailed test) P
8	iwk	I*	3×n	Work	Work area
9	wk	{D∗ / R∗}	2×n	Work	Work area
10	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) n≥2

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
1000	Samples A and B are equal (izr=0).	t=0 and p=0 are set, and the absolute value maximum that can be represented is set for z.
3000	Restriction (a) was not satisfied.	Processing is aborted.

(6) **Notes**

(a) The number of observed values in the two samples must be equal.

(b) When this function is used, the test is a one-tailed test. However, when this procedure is used for a two-tailed test, if the significance level is α, the null hypothesis is rejected when the following relationship holds.

$$P \leq \frac{\alpha}{2}$$

(7) **Example**

(a) Problem

Perform a Wilcoxon test for the following set of observed values of two samples.

$$\begin{aligned}\{(x_i, y_i)\} &= \{(28,36),(96,24),(37,47),(34,73),(85,15),(56,34),(67,80),\\ &\quad (56,82),(19,78),(27,41),(61,56),(76,32),(13,31),(43,41)\}\end{aligned}$$

(b) Input data

Set of observed values $\{(x_i, y_i)\}$ and n=14.

(c) Main program

```
/*      C interface example for ASL_d5tewl */

#include <stdio.h>
#include <stdlib.h>
#include <asl.h>

int main()
{
    double *a;
    int n;
    double *b;
    int izr;
    double t;
    double z;
    double p;
    int *iwk;
    double *wk;
    int ierr;
    int i;
    FILE *fp;

    fp = fopen( "d5tewl.dat", "r" );
    if( fp == NULL )
    {
        printf( "file open error\n" );
        return -1;
    }

    printf( "    *** ASL_d5tewl ***\n" );
    printf( "\n     ** Input **\n\n" );

    fscanf( fp, "%d", &n );

    a = ( double * )malloc((size_t)( sizeof(double) * n ));
    if( a == NULL )
    {
        printf( "no enough memory for array a\n" );
        return -1;
    }

    b = ( double * )malloc((size_t)( sizeof(double) * n ));
    if( b == NULL )
    {
        printf( "no enough memory for array b\n" );
        return -1;
    }

    iwk = ( int * )malloc((size_t)( sizeof(int) * (3*n) ));
    if( iwk == NULL )
    {
        printf( "no enough memory for array iwk\n" );
        return -1;
    }

    wk = ( double * )malloc((size_t)( sizeof(double) * (2*n) ));
    if( wk == NULL )
    {
```

```
        printf( "no enough memory for array wk\n" );
        return -1;
    }

    printf( "\tn = %6d\n", n );

    printf("\n\tObservations\n\n");
    printf("\t    No.       A        B\n");
    printf("\t------------------------------\n");
    for( i=0 ; i<n ; i++ )
    {
        fscanf( fp, "%lf %lf", &a[i], &b[i] );
        printf( "\t%6d %8.3g %8.3g\n", i+1, a[i], b[i] );
    }

    fclose( fp );

    ierr = ASL_d5tewl(a, n, b, &izr, &t, &z, &p, iwk, wk);

    printf( "\n     ** Output **\n\n" );
    printf( "\tierr = %6d\n", ierr );
    printf( "\n\tNumber of pairs = %6d\n", izr );
    printf( "\n\tStatistical value = %8.3g\n", t );
    printf( "\n\tNormalized value  = %8.3g\n", z );
    printf( "\n\tProbability = %8.3g not significant at 0.05 level\n", p );

    free( a );
    free( b );
    free( iwk );
    free( wk );

    return 0;
}
```

(d) Output results

```
*** ASL_d5tewl ***

 ** Input **

n =     14

Observations

    No.        A         B
------------------------------
     1       28        36
     2       96        24
     3       37        47
     4       34        73
     5       85        15
     6       56        34
     7       67        80
     8       56        82
     9       19        78
    10       27        41
    11       61        56
    12       76        32
    13       13        31
    14       43        41

 ** Output **

ierr =      0

Number of pairs =     14

Statistical value =       49

Normalized value  =    -0.22

Probability =    0.413 not significant at 0.05 level
```

8.3.3 ASL_d5temh, ASL_r5temh Mann-Whitney's U Test

(1) **Function**

The ASL_d5temh or ASL_r5temh performs Mann-Whitney's U test for two independent samples.
Given n observed values x_i, $(i = 1, \cdots, n)$ and m observed values y_j, $(j = 1, \cdots, m)$, which were independently taken from two populations $\prod_1$ and $\prod_2$ having continuous distribution functions $F_1(x)$ and $F_2(x)$, this function tests the null hypothesis H_0: "$F_1(x) = F_2(x)$" against the alternative hypothesis H_1: "$F_1(x) > F_2(x)$" or H_1: "$F_1(x) < F_2(x)$." The value for the n observed values x_i, $(i = 1, \cdots, n)$ and m observed values y_i, $(j = 1, \cdots, m)$ (where $n \leq m$) is defined by the equations shown below.
The following kinds of values R_i and T are obtained from the observed values.

R_i: Ranks assigned to $(n + m)$ observed values obtained by combining x_i and y_i. The mean rank is assigned for equal ranks.

T: Sum of ranks assigned to x_i

Test statistic :

$$U = \min(U_1, U_2)$$

Here, U_1 and U_2 are defined as follows.

$$U_1 = n \cdot m + \frac{n(n+1)}{2} - T$$

$$U_2 = n \cdot m - U_1$$

Expected value of U :

$$E[U] = \frac{n \cdot m}{2}$$

Variance of U:

- No equal ranks exist

$$V[U] = \frac{n \cdot m(n+m+1)}{12}$$

- Equal ranks exist

$$V[U] = \frac{n \cdot m}{12(n+m)(n+m-1)}((n+m)^3 - (n+m) - \sum S)$$

Here, S is defined as follows.

$$S = \sum(t^3 - t)$$

t : Number of values having equal ranks according to assigned ranks

Probability :

$$P = \int_{-\infty}^{Z} \frac{1}{\sqrt{2 \cdot \pi}} e^{-\frac{x^2}{2}} dx$$

Here, Z is defined as follows.

$$Z = \frac{U - E[U]}{\sqrt{V[U]}}$$

(The Z for which U has been normalized obeys a standard normal distribution $N(0, 1)$.)

(2) **Usage**

Double precision:

```
ierr = ASL_d5temh  (a, n, m, r, &u, &z, &p, iwk);
```

Single precision:

```
ierr = ASL_r5temh  (a, n, m, r, &u, &z, &p, iwk);
```

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	a	{D∗ / R∗}	n+m	Input	Of the two independent samples, enter the observed values of the one having fewer observed values first, followed by the observed values of the one having more observed values.
2	n	I	1	Input	Number of observed values in the sample having fewer observed values n
3	m	I	1	Input	Number of observed values in the sample having more observed values m
4	r	{D∗ / R∗}	n+m	Output	Ranks assigned to observed values obtained by combining the two samples $\{R_i\}$
5	u	{D∗ / R∗}	1	Output	Test statistic U
6	z	{D∗ / R∗}	1	Output	Value for computing the significance of u according to a normal distribution (value for which u has been normalized) Z
7	p	{D∗ / R∗}	1	Output	Probability (one-tailed test) P
8	iwk	I*	See Contents	Work	Work area **Size:** 3×(n+m)
9	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) n≥1

(b) m≥20

(c) n≤m

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
3000	Any of restriction (a) to (c) were not satisfied.	Processing is aborted.

(6) **Notes**

(a) When this function is used, the test is a one-tailed test. However, when this procedure is used for a two-tailed test, if the significance level is α, the null hypothesis is rejected when the following relationship holds.

$$P \leq \frac{\alpha}{2}$$

(7) **Example**

(a) Problem
Perform Mann-Whitney's U test for the two independent samples given by the following observed values.

$$\{x_i\} = \{13, 12, 12, 10, 10, 10, 10, 9, 8, 8, 7, 7, 7, 7, 7, 6\}$$

and

$$\{y_i\} = \{17, 16, 15, 15, 15, 14, 14, 14, 13, 13, 13, 12, 12, 12, 12, 11, 11, 10, 10, 10, 8, 8, 6\}$$

(b) Input data
Observed values $\{x_i\}$, n=16, observed values $\{y_i\}$ and m=23.

(c) Main program

```
/*      C interface example for ASL_d5temh */

#include <stdio.h>
#include <stdlib.h>
#include <asl.h>

int main()
{
    double *a;
    int n;
    int m;
    double *r;
    double u;
    double z;
    double p;
    int *iwk;
    int ierr;
    int i;
    FILE *fp;

    fp = fopen( "d5temh.dat", "r" );
    if( fp == NULL )
    {
        printf( "file open error\n" );
        return -1;
    }

    printf( "    *** ASL_d5temh ***\n" );
    printf( "\n     ** Input **\n\n" );

    fscanf( fp, "%d", &n );
    fscanf( fp, "%d", &m );

    a = ( double * )malloc((size_t)( sizeof(double) * (n+m) ));
    if( a == NULL )
    {
        printf( "no enough memory for array a\n" );
        return -1;
    }

    r = ( double * )malloc((size_t)( sizeof(double) * (n+m) ));
```

```
    if( r == NULL )
    {
        printf( "no enough memory for array r\n" );
        return -1;
    }

    iwk = ( int * )malloc((size_t)( sizeof(int) * 3 * (n+m) ));
    if( iwk == NULL )
    {
        printf( "no enough memory for array iwk\n" );
        return -1;
    }

    printf( "\tn=%3d,  m=%3d\n", n, m );

    printf("\n\tObservations A\n");
    for( i=0 ; i<n ; i++ )
    {
        if( i%5 == 0 )
        {
            printf( "\n\t" );
        }
        fscanf( fp, "%lf", &a[i] );
        printf( "%8.3g", a[i] );
    }
    printf( "\n" );

    printf("\n\tObservations B\n");
    for( i=n ; i<n+m ; i++ )
    {
        if( (i-n)%5 == 0 )
        {
            printf( "\n\t" );
        }
        fscanf( fp, "%lf", &a[i] );
        printf( "%8.3g", a[i] );
    }
    printf( "\n" );

    fclose( fp );

    ierr = ASL_d5temh(a, n, m, r, &u, &z, &p, iwk);

    printf( "\n     ** Output **\n\n" );
    printf( "\tierr = %6d\n", ierr );
    printf( "\n\tU-value          = %8.3g\n", u );
    printf( "\n\tNormalized value = %8.3g\n", z );
    printf( "\n\tProbability = %8.3g significant at 0.05 level\n", p );

    free( a );
    free( r );
    free( iwk );

    return 0;
}
```

(d) Output results

```
*** ASL_d5temh ***

 ** Input **

n= 16,  m= 23

Observations A

      13      12      12      10      10
      10      10       9       8       8
       7       7       7       7       7
       6

Observations B

      17      16      15      15      15
      14      14      14      13      13
      13      12      12      12      12
      11      11      10      10      10
       8       8       6

 ** Output **

ierr =      0

U-value          =       64

Normalized value =    -3.45

Probability = 0.000279 significant at 0.05 level
```

8.3.4 ASL_d5tesp, ASL_r5tesp Spearman's Rank Correlation Test

(1) **Function**

The ASL_d5tesp or ASL_r5tesp obtains Spearman's rank correlation coefficient and tests the correlation between two samples.

When (X, Y) is assumed to be a pair of random variables and n pairs of observed values $(x_i, y_i),\ (i = 1, \cdots, n)$ are given as the actual values for them, this function tests the null hypothesis H_0: "X and Y are independent" against the alternative hypothesis H_1: "There is a positive correlation between X and Y" or H_1: "There is a negative correlation between X and Y."

The value for the n pairs of observed values $(x_i, y_i),\ (i = 1, \cdots, n)$ is defined by the equations shown below. First, the two samples are ranked individually and represented by a_i and b_i, respectively. The mean rank is assigned for equal ranks.

Spearman's rank correlation coefficient :

- No equal ranks exist

$$r_s = 1 - \frac{6\sum_{i=1}^{n}(a_i - b_i)^2}{n^3 - n}$$

- Equal ranks exist

$$r_s = \frac{(n^3 - n - S_1) + (n^3 - n - S_2) - 12\sum_{i=1}^{n}(a_i - b_i)^2}{2\sqrt{(n^3 - n - S_1)(n^3 - n - S_2)}}$$

Here, S_1, which is the correction factor of the first sample, and S_2, which is the correction factor of the second sample, are defined as follows.

$$S_1 = \sum({t_1}^3 - t_1)$$

$$S_2 = \sum({t_2}^3 - t_2)$$

t : Number of values having equal ranks according to assigned ranks

Test statistic :

$$T = r_s\sqrt{\frac{n-2}{1-{r_s}^2}}$$

(The T obeys a t distribution with $n - 2$ degrees of freedom.)

(2) **Usage**

Double precision:

```
ierr = ASL_d5tesp  (a, n, b, &idf, r1, r2, &rs, &t, iwk);
```

Single precision:

```
ierr = ASL_r5tesp  (a, n, b, &idf, r1, r2, &rs, &t, iwk);
```

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	a	{D∗ / R∗}	n	Input	Observed values of sample A $\{x_i\}$
2	n	I	1	Input	Number of observed values in each of the two samples n
3	b	{D∗ / R∗}	n	Input	Observed values of sample B $\{y_i\}$
4	idf	I*	1	Output	Degrees of freedom
5	r1	{D∗ / R∗}	n	Output	Ranks assigned to observed values of sample A $\{a_i\}$
6	r2	{D∗ / R∗}	n	Output	Ranks assigned to observed values of sample B $\{b_i\}$
7	rs	{D∗ / R∗}	1	Output	Spearman's rank correlation coefficient r_s
8	t	{D∗ / R∗}	1	Output	Value for computing the significance of rs according to a t distribution T
9	iwk	I*	3×n	Work	Work area
10	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) n≥10

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
3000	Restriction (a) was not satisfied.	Processing is aborted.
4000	rs was -1 or 1.	

(6) **Notes**

(a) The number of observed values in the two samples must be equal.

(b) The rankings r1 and r2 correspond to the samples A and B, respectively.

(c) When this function is used, the test is a one-tailed test. However, when this procedure is used for a two-tailed test, if the number of degrees of freedom is n and the significance level is α, he null hypothesis is rejected when the either of the following relationships holds, based on the value of $t_0\left(n, \frac{\alpha}{2}\right)$:

$$t \geq t_0\left(n, \frac{\alpha}{2}\right)$$

or

$$t \leq -t_0\left(n, \frac{\alpha}{2}\right)$$

(7) **Example**

(a) Problem

Obtain Spearman's rank correlation coefficient and test the correlation between the two samples for the following set of observed values of two samples.

$$\begin{aligned}\{(x_i, y_i)\} &= \{(93,53),(98,46),(76,28),(28,25),(103,65),(99,80),(98,73),\\ &\quad (71,44),(74,51),(116,82),(97,54)\}\end{aligned}$$

(b) Input data

Set of observed values $\{(x_i, y_i)\}$ and n=11.

(c) Main program

```
/*      C interface example for ASL_d5tesp */

#include <stdio.h>
#include <stdlib.h>
#include <asl.h>

int main()
{
    double *a;
    int n;
    double *b;
    int idf;
    double *r1;
    double *r2;
    double rs;
    double t;
    int *iwk;
    int ierr;
    int i;
    FILE *fp;

    fp = fopen( "d5tesp.dat", "r" );
    if( fp == NULL )
    {
        printf( "file open error\n" );
        return -1;
    }

    printf( "    *** ASL_d5tesp ***\n" );
    printf( "\n     ** Input **\n\n" );

    fscanf( fp, "%d", &n );

    a = ( double * )malloc((size_t)( sizeof(double) * n ));
    if( a == NULL )
    {
        printf( "no enough memory for array a\n" );
        return -1;
    }

    b = ( double * )malloc((size_t)( sizeof(double) * n ));
    if( b == NULL )
    {
        printf( "no enough memory for array b\n" );
        return -1;
    }
```

```
    r1 = ( double * )malloc((size_t)( sizeof(double) * n ));
    if( r1 == NULL )
    {
        printf( "no enough memory for array r1\n" );
        return -1;
    }

    r2 = ( double * )malloc((size_t)( sizeof(double) * n ));
    if( r2 == NULL )
    {
        printf( "no enough memory for array r2\n" );
        return -1;
    }

    iwk = ( int * )malloc((size_t)( sizeof(int) * (3*n) ));
    if( iwk == NULL )
    {
        printf( "no enough memory for array iwk\n" );
        return -1;
    }

    printf( "\tn = %6d\n", n );
    printf("\n\tObservations\n\n");
    printf("\t    No.       A        B\n");
    printf("\t------------------------------\n");
    for( i=0 ; i<n ; i++ )
    {
        fscanf( fp, "%lf %lf", &a[i], &b[i] );
        printf( "\t%6d %8.3g %8.3g\n", i+1, a[i], b[i] );
    }

    fclose( fp );

    ierr = ASL_d5tesp(a, n, b, &idf, r1, r2, &rs, &t, iwk);

    printf( "\n     ** Output **\n\n" );
    printf( "\tierr = %6d\n", ierr );

    printf( "\n\tn = %6d\n", n );
    printf( "\n\tTied ranked\n\n" );
    printf("\t    No.       A        B\n");
    printf("\t------------------------------\n");
    for( i=0 ; i<n ; i++ )
    {
        printf( "\t%6d %8.3g %8.3g\n", i+1, r1[i], r2[i] );
    }

    printf( "\n\tDegree of freedom = %6d\n", idf );
    printf( "\n\tSpearman rank correlation coefficient = %8.3g\n", rs );
    printf( "\n\tStatistical value = %8.3g significant at 0.05 level\n", t);

    free( a );
    free( b );
    free( r1 );
    free( r2 );
    free( iwk );

    return 0;
}
```

(d) Output results

```
*** ASL_d5tesp ***

 ** Input **

n =     11

Observations

    No.        A         B
------------------------------
     1       93        53
     2       98        46
     3       76        28
     4       28        25
     5      103        65
     6       99        80
     7       98        73
     8       71        44
     9       74        51
    10      116        82
    11       97        54

 ** Output **

ierr =      0

n =     11

Tied ranked

    No.        A         B
------------------------------
     1        5         6
     2      7.5         4
     3        4         2
     4        1         1
     5       10         8
     6        9        10
     7      7.5         9
     8        2         3
     9        3         5
    10       11        11
    11        6         7

Degree of freedom =      9

Spearman rank correlation coefficient =    0.861

Statistical value =     5.08 significant at 0.05 level
```

Chapter 9

MULTIVARIATE ANALYSIS

9.1 INTRODUCTION

When several sets of various individuals are observed, an analysis of that data is called **multivariate analysis**. This library provides the following functions for performing a multivariate analysis.

- Principal Component Analysis
- Factor Analysis
- Canonical Correlation Analysis
- Discriminant Analysis
- Cluster Analysis

9.1.1 Explanation

(1) **Principal Component Analysis**

The objective of a principal component analysis is to explain numerous variates according to a small number of variates. Let the mean vector and variance-covariance matrix of the probability vector $\boldsymbol{x} = [x_1, \cdots, x_n]^T$ be represented by $\boldsymbol{\mu}$ and Σ, respectively. Consider the following linear combination of $\boldsymbol{x}$ according to the vector $\boldsymbol{c}$.

$$y = \boldsymbol{c}^T(\boldsymbol{x} - \boldsymbol{\mu}), \boldsymbol{c}^T\boldsymbol{c} = 1$$

The n variates $y_i = \boldsymbol{c_i}^T(\boldsymbol{x} - \boldsymbol{\mu})$ that satisfy the following conditions :

- y_1 maximizes the variance of y in relation to $\boldsymbol{c}$.
- When $y_1, \cdots, y_k$ are defined for $k < n$, $y_{k+1} = \boldsymbol{c_{k+1}}^T(\boldsymbol{x} - \boldsymbol{\mu})$ maximizes the variance of y in relation to $\boldsymbol{c}$ based on $Cov(y, y_i) = 0, i = 1, \cdots, k$.

are called the ith principal components of x, and the $\boldsymbol{c_i}$ are called the principal component vectors. The $\boldsymbol{c_i}$ also constitute the orthonormal eigenvectors of Σ. That is, if the eigenvalues of Σ are given by $\lambda_1 \leq \cdots \leq \lambda_n$, the corresponding orthonormal eigenvector is $C = [\boldsymbol{c_1}, \cdots, \boldsymbol{c_n}]$. The contribution ratio of the ith principal component, which is defined as follows:

$$\frac{\lambda_i}{\sum_{i=1}^n \lambda_i}$$

represents the proportion of the total variation of the variance of y_i. Also, the cumulative contribution ratio up to the kth principal component is given as follows.

$$\frac{\sum_{i=1}^k \lambda_i}{\sum_{i=1}^n \lambda_i}$$

The principal component score vector of individual j is defined as follows.

$$\boldsymbol{y_j} = C^T(\boldsymbol{x_j} - \boldsymbol{\mu})$$

Generally, since $\boldsymbol{\mu}$ and Σ are unknown, the sample mean vector and sample variance-covariance matrix are used as estimates instead. Also, since the principal components are not invariant based on a scaling transformation, in practice, the principal component analysis is performed for a standardized variate in place of $\boldsymbol{x}$.

(2) **Factor Analysis**

The objective of a factor analysis is to explain the correlation relationships among several variables according to "factors" (which are fewer in number than the number of variables). Let the observed values of m variates x_i $(i = 1, 2, \cdots, m)$ consisting of n observed values be represented by $x_{i,j}$ $(i = 1, 2, \cdots, m;\ j = 1, 2, \cdots, n)$, and let the mean and variance of each variate be represented by $\overline{x_i}$ and σ_i^2, respectively. Let the vector $\boldsymbol{z}_j = [z_{1,j}, z_{2,j}, \cdots, z_{m,j}]^T$ consist of the standardized variables $z_{i,j}$ that correspond to each variate, where the $z_{i,j}$ are defined as follows:

$$z_{i,j} = \frac{x_{i,j} - \overline{x_i}}{\sigma_i}$$

Assume that the vector $\boldsymbol{z}_j$ has the following kind of structure.

$$\boldsymbol{z}_j = F\boldsymbol{f}_j + \boldsymbol{u}_j$$

Here, $F = (a_{i,k})$ $(i = 1, 2, \cdots, m;\ k = 1, 2, \cdots, l)$ $\boldsymbol{f}_j = [f_{1,j}, f_{2,j}, \cdots, f_{l,j}]^T$, and $\boldsymbol{u}_j = [u_{1,j}, u_{2,j}, \cdots, u_{l,j}]^T$ represent the following values, respectively:

- $a_{i,k}$: Factor loading (unknown constant) related to the kth factor of the ith variable
- $f_{k,j}$: kth common factor score of jth observed value
- $u_{k,j}$: Characteristic factor score related to the kth variable of the jth observed value

At this time, the correlation coefficient matrix between the standardized variables R can be decomposed as follows.

$$R = FF^T + D = R^* + D$$

Here, D is a diagonal matrix whose ith principal diagonal term element is called the **specificity** related to the ith variable. The ith principal diagonal term element of R^* is called the **communality** of the ith variable. Let the eigenvalues of R^* be λ_i, and let the corresponding eigenvectors be $W = [\boldsymbol{w}_1, \cdots, \boldsymbol{w}_m] = (w_{i,j})$. To form an l factor model, represent the **factor matrix** F as follows.

$$F = [\sqrt{\lambda_1}\boldsymbol{w}_1, \cdots, \sqrt{\lambda_l}\boldsymbol{w}_l]$$

The **factor scores** $H = [\boldsymbol{f}_1, \boldsymbol{f}_2, \cdots, \boldsymbol{f}_n]^T$ are obtained by solving the following equations.

$$\begin{aligned} H &= ZW \\ (F^TF)W &= F \end{aligned}$$

Here, $Z = [\boldsymbol{z}_1, \boldsymbol{z}_2, \cdots, \boldsymbol{z}_n]^T$. Generally, the factor matrix does not take a suitable form for interpreting the actual contents of each factor. Therefore, a method is considered of applying a suitable orthogonal rotation (**factor orthogonal rotation**) to the factor matrix to make the results easier to interpret. When performing an orthogonal rotation of the factor matrix, you need to determine the kind of factor structure to aim for when performing the rotation. A representative example of an orthogonal rotation criterion is the **varimax criterion**. With the **varimax method**, the factor matrix is orthogonally rotated so that the sum for all factors of the squared variance of the factor loading of each factor (varimax criterion) is maximized. The procedure is as follows.

① Obtain the communality before rotation.

$$h_i^2 = \sum_{j=1}^{k} a_{ij}^2$$

$(i = 1, 2, \cdots, m; (\text{Number of variates}),\quad j = 1, 2, \cdots, k; (\text{Number of factors}))$

$A = (a_{ij})$; Factor loading matrix

② Normalize the factor loading matrix.

$$b_{ij} = a_{ij}/\sqrt{h_i^2}\ \ (i = 1, 2, \cdots, m;\quad j = 1, 2, \cdots, k)$$

③ Perform an orthogonal rotation that maximizes the variance of the factor loading matrix, which is shown below.

$$V_c = \sum_{j=1}^{k} \frac{m\sum_{i=1}^{m}(b_{ij}^2)^2 - (\sum_{i=1}^{m} b_{ij}^2)^2}{m^2}\ \ (c = 1, 2, \cdots, r(\text{Maximum number of iterations}))$$

Let the transformation matrix $T_{jj'}$ for the pair (j, j') among k factors be as follows.

$$T_{jj'} = \begin{array}{c} \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \end{array} \begin{array}{c} \begin{array}{ccccccccccc} & & & \text{j}-\text{th column} & & & & \text{j}'-\text{th column} & & & \end{array} \\ \begin{bmatrix} 1 & & & \vdots & & & & \vdots & & & \\ & \ddots & & \vdots & & & & \vdots & & 0 & \\ & & 1 & \vdots & & & & \vdots & & & \\ \cdots & \cdots & \cdots & \cos\theta & \cdots & \cdots & \cdots & -\sin\theta & \cdots & \cdots & \cdots \\ & & & \vdots & 1 & & & \vdots & & & \\ & & & \vdots & & \ddots & & \vdots & & & \\ & & & \vdots & & & 1 & \vdots & & & \\ \cdots & \cdots & \cdots & \sin\theta & \cdots & \cdots & \cdots & \cos\theta & \cdots & \cdots & \cdots \\ & & & \vdots & & & & \vdots & 1 & & \\ & 0 & & \vdots & & & & \vdots & & \ddots & \\ & & & \vdots & & & & \vdots & & & 1 \end{bmatrix} \end{array} \begin{array}{l} \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \end{array} \begin{array}{l} \\ \\ \\ \text{j}-\text{th row} \\ \\ \\ \\ \\ \text{j}'-\text{th row} \\ \\ \\ \end{array}$$

At this time, let u_i, v_i, p, q, r and s be defined as follows.

$$u_i = a_{ij}/h_i, \quad v_i = a_{ij'}/h_i$$

$$p = \sum_{i=1}^{m}(u_i^2 - v_i^2)$$

$$q = 2\sum_{i=1}^{m} u_i v_i$$

$$r = \sum_{i=1}^{m}\{(u_i^2 - v_i^2)^2 - 4u_i^2 v_i^2\}$$

$$s = 4\sum_{i=1}^{m} u_i v_i (u_i^2 - v_i^2)$$

Then, θ is given by the following equation.

$$\theta = \frac{1}{4}\tan^{-1}\frac{s - \dfrac{pq}{m}}{r - \dfrac{p^2 - q^2}{m}} \quad (-\frac{\pi}{4} \le \theta \le \frac{\pi}{4})$$

The rotation for the combination of all factors is given as follows:

$$AT_{1,2}T_{1,3}\cdots T_{1,k}T_{2,3}\cdots T_{2,k}\cdots T_{k-1,k}$$

and V_c is calculated at this time. This value is compared with V_{c-1} of the previous time, and the calculation ends if the following condition is satisfied.

$$|V_c - V_{c-1}| \le \varepsilon$$

If this condition is not satisfied, rotation is repeated. Here, ε is 10^{-7} for single precision and 10^{-12} for double precision. Also c denotes number of rotation iterations.

④ When the final V_c is calculated, the difference of the communality for the factor matrix after rotation, which is denoted by f_i as follows :

$$f_i = \sum_{j=1}^{k} a_{ij}^2 \quad (i = 1, 2, \cdots, m)$$

and the communality before rotation is calculated as follows.

$$d_i = h_i^2 - f_i$$

(3) **Canonical Correlation Analysis**

When observed values $x_{i,j}$ $(i = 1, 2, \cdots, n;\ j = 1, 2, \cdots, l+m)$ for $l+m$ variables x_j $(j = 1, 2, \cdots, l+m)$ concerning n subjects are given, a canonical correlation analysis is a method in which these $l+m$ variables are divided into two groups according to some criteria, and the relationships between these two groups are analyzed according to the canonical correlation coefficients. Let the mean and variance of the observed values for each variable be represented by $\overline{x_j}$ and σ_j^2, respectively. The standardized results $u_{i,j}$ corresponding to each variable are defined as follows.

$$u_{i,j} = \frac{x_{i,j} - \overline{x_j}}{\sigma_j}$$

Now, assume that the (standardized) observed values are divided into the following two groups.
First group: $u_{i,j}(i = 1, 2, \cdots, n; j = 1, 2, \cdots, l)$
Second group: $u_{i,j}(i = 1, 2, \cdots, n; j = l+1, 2, \cdots, l+m)$
At this time, consider the composite variables z and w determined as follows by the values.

$$z_i = \sum_{j=1}^{l} p_i u_{i,j}$$
$$w_i = \sum_{j=l+1}^{l+m} q_i u_{i,j}$$

The coefficients p_i and q_i are determined so that the values of the variances σ_z^2 and σ_w^2 of the composite variables z and w, respectively, are 1. At this time, the composite variables z and w are called **canonical variates**, and the correlation coefficients ρ_{zw} of z and w are called the **canonical correlation coefficients**. Since $\sigma_z^2 = \sigma_w^2 = 1$, the following relationship holds.

$$\rho_{zw} = \sigma_{zw} = \sum_{i=1}^{n} \frac{z_i w_i}{n}$$

Here, σ_{zw} is the covariance of the composite variables z and w. A canonical correlation analysis determines the coefficients $\boldsymbol{p} = \{p_i\}$ and $\boldsymbol{q} = \{q_i\}$ so that the values of the canonical correlation coefficients are maximized and views the strength of the relationship between the two groups that were divided according to the canonical correlation coefficients at that time. Now, let the correlation coefficient matrix of the first group be S (size: $l \times l$), the correlation coefficient matrix of the second group be T (size: $m \times m$), and the correlation coefficient matrix of the first and second groups be R (size: $l \times m$). For indeterminate constants λ and μ the following equations are obtained by using Lagrange's method of indeterminate coefficients.

$$R\boldsymbol{q} = \lambda S\boldsymbol{p}$$
$$R^T\boldsymbol{p} = \mu T\boldsymbol{q}$$

Now, from the following relationships:

$$\sigma_z^2 = \boldsymbol{p}^T S\boldsymbol{p} = 1$$
$$\sigma_w^2 = \boldsymbol{q}^T T\boldsymbol{q} = 1$$

we find that:

$$\begin{aligned} \lambda = \mu = \rho_{zw} \\ S^{-1}RT^{-1}R^T\boldsymbol{p} = \lambda^2\boldsymbol{p} \\ \boldsymbol{q} = \lambda^{-1}T^{-1}R^T\boldsymbol{p} \end{aligned}$$

and λ, that is ρ_{zw}, is obtained from the maximum eigenvalues of the matrix $S^{-1}RT^{-1}R^T$. Now, $\boldsymbol{p}$ and $\boldsymbol{q}$ should be determined so that the following relationships are satisfied among the corresponding eigenvectors.

$$\begin{aligned} \boldsymbol{p}^T S\boldsymbol{p} = 1 \\ \boldsymbol{q}^T T\boldsymbol{q} = 1 \end{aligned}$$

The number of nonzero canonical correlation coefficients is said to be the number of dimensions of a canonical variate. The number of dimensions can be determined by sequentially performing hypothesis tests that test the following null hypothesis:

$$H_k : \lambda_{k+1} = \cdots = \lambda_l = 0$$

against the following alternative hypothesis:

$$K_k : H_k \text{ is not true}$$

That is, if $H_0, \cdots, H_{k-1}$ are rejected and H_k is adopted, the number of dimensions is assumed to be k. The test is performed by using the following fact. If Wilks' Λ, which is defined by the following equation, is used:

$$\Lambda_k = \prod_{i=k+1}^{l} (1 - {\lambda_i}^2)$$

based on hypothesis H_k, the following χ_k^2 values:

$$\chi_k^2 = -\{n - 0.5(l + m + 1)\} \log_e \Lambda_k$$

asymptotically obey a χ^2 distribution with $(l - k)(m - k)$ degrees of freedom.

(4) **Discriminant Analysis**

A discriminant analysis deals with the problem of discriminating the population to which a certain individual belongs among the k populations $\pi_1, \cdots, \pi_k$ based on observed values for that individual. A prerequisite of this problem is that the individual for which the population is to be discriminated belongs to some population among $\pi_1, \cdots, \pi_k$. When the various populations are normal populations of order p represented by $N(\boldsymbol{\nu_1}, \Sigma), \cdots, N(\boldsymbol{\nu_k}, \Sigma)$, the following kind of linear function (**linear discriminant function**):

$$y^{(i)}(\boldsymbol{x}) = {\boldsymbol{\nu_i}}^T \Sigma^{-1} \boldsymbol{x} - \frac{1}{2} {\boldsymbol{\nu_i}}^T \Sigma^{-1} \boldsymbol{\nu_i}$$

can be used to discriminate:

$$\max_j y^j(\boldsymbol{x}) = y^{j_m}(\boldsymbol{x}) \Rightarrow \boldsymbol{x} \in \pi_{j_m}$$

When the parameters are unknown, these estimates can be used for discrimination.

(5) **Cluster Analysis**

From a statistical data analysis viewpoint, classification means to provide a scale for representing the similarities and differences seen among the classification subjects and dividing those subjects into several groups (clusters) according to that scale. With this meaning, classification can be called a **clustering method** or **cluster generation method**. Cluster analysis uses an (individual)× (variate) multivariate characteristic value data matrix for the data handled as the classification subjects. At this time, both the individuals and the variates are treated as classification subjects. In either case, the cluster generation process requires a measure for representing the similarities or differences of the classification subjects. This measure is called the **similarity measure** or **dissimilarity measure**. If individuals or variates having n characteristics are to be used as classification subjects when the (individual)× (variate) multivariate characteristic value data matrix (a_{ik}) or (a_{ki}) $(i = 1, 2, \cdots, n; k = 1, 2, \cdots, p)$ is given, the following measures, for example, can be used as the dissimilarity measure d_{ij} $(i, j = 1, 2, \cdots, n)$.

- Euclidean quadratic distance

$$d_{ij} = \sum_{k=1}^{p} (a_{ik} - a_{jk})^2 \ (i, j = 1, \cdots, n)$$

- Standardized Euclidean quadratic distance

$$d_{ij} = \sum_{k=1}^{p} \frac{(a_{ik} - a_{jk})^2}{s_k^2} \ (i, j = 1, \cdots, n)$$

Here, s_k^2, which is the variance of the variate, is defined by the following equation.

$$s_k^2 = \frac{1}{n-1} \sum_{l=1}^{n} (a_{lk} - \bar{a_k})^2 (\bar{a_k} = \frac{1}{n} \sum_{l=1}^{n} a_{lk})$$

This is the same as obtaining the Euclidean quadratic distance when the variance of each variate is standardized to 1.

- Generalized distance of Mahalanobis

$$d_{ij} = \sum_{k=1}^{p} \sum_{m=1}^{p} (a_{ik} - a_{jk}) v_{km} (a_{im} - a_{jm}) \ (i, j = 1, \cdots, n)$$

Here, v_{km} is the (k, m) element of the inverse matrix of the variance-covariance matrix of the individuals and variates.

- Minkowski distance

$$d_{ij} = \{ \sum_{k=1}^{p} \mid a_{ik} - a_{jk} \mid^r \}^{1/r} \ (r \geq 1.0; i, j = 1, \cdots, n)$$

Also, to perform the classification, a measure must be defined for use when individuals or variates are merged as a cluster. When a new cluster t is created by merging cluster p and cluster q, the following measures are used as the dissimilarity measure d_{tr} between cluster t and a separate arbitrary cluster r. Here, n_p represents the number of subjects contained in cluster p.

- Nearest neighbor method

$$d_{tr} = \min(d_{pr}, d_{qr})$$

- Furtherest neighbor method

$$d_{tr} = \max(d_{pr}, d_{qr})$$

- Group mean method

$$d_{tr} = (n_p d_{pr} + n_q d_{qr})/(n_p + n_q)$$

- Center of gravity method

$$d_{tr} = \frac{n_p}{n_p + n_q} d_{pr} + \frac{n_q}{n_p + n_q} d_{qr} - \frac{n_p n_q}{(n_p + n_q)^2} d_{pq}$$

- Median method

$$d_{tr} = \frac{1}{2} d_{pr} + \frac{1}{2} d_{qr} - \frac{1}{4} d_{pq}$$

- Ward's method

$$d_{tr} = \frac{n_p + n_r}{n_t + n_r} d_{pr} + \frac{n_q + n_r}{n_t + n_r} d_{qr} - \frac{n_r}{n_t + n_r} d_{pq}$$

- Variable method

$$d_{tr} = \frac{1 - \beta}{2} d_{pr} + \frac{1 - \beta}{2} d_{qr} + \beta d_{pq} \ (-\frac{1}{4} \leq \beta \leq 0)$$

The center of gravity method, median method, and Ward's method assume that the dissimilarity measure is given by the (standardized) Euclidean quadratic distance.

9.1.2 Reference Bibliography

(1) Anderson, T. W. , "An Introduction to Multivariate Statistical Analysis", John Wiley & Sons, New York (1958)

9.2 PRINCIPAL COMPONENT ANALYSIS

9.2.1 ASL_d6cpcc, ASL_r6cpcc
Principal Component Cumulative Contribution Ratio

(1) **Function**

Let the mean vector and variance-covariance matrix of the probability vector $\boldsymbol{x} = [x_1, \cdots, x_m]^T$ consisting of m variates be represented by $\boldsymbol{\mu}$ and Σ, respectively. If the eigenvalues of Σ are given by $\lambda_1 \leq \cdots \leq \lambda_m$, and the corresponding orthonormal eigenvector is given by $C = [\boldsymbol{c_1}, \cdots, \boldsymbol{c_m}]$, the contribution ratio of the ith principal component, which is defined as follows:

$$\frac{\lambda_i}{\sum_{i=1}^{m} \lambda_i}$$

represents the proportion of the total variation of the variance of y_i. Also, the cumulative contribution ratio up to the kth principal component is given as follows.

$$c_k = \frac{\sum_{i=1}^{k} \lambda_i}{\sum_{i=1}^{m} \lambda_i}$$

When the eigenvalues $\lambda_1 \leq \cdots \leq \lambda_m$ are given, and the given criterion value is represented by s, the ASL_d6cpcc or ASL_r6cpcc obtains the minimum $k = k_m$ value and the cumulative contribution ratio c_k $(k = 1, 2, \cdots, k_m)$ that satisfies the following relationship.

$$c_k \geq s$$

(2) **Usage**

Double precision:

```
ierr = ASL_d6cpcc  (a, m, cons, cp, &num);
```

Single precision:

```
ierr = ASL_r6cpcc  (a, m, cons, cp, &num);
```

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	a	{D∗ / R∗}	m	Input	Eigenvalues λ_i (See Note (a))
2	m	I	1	Input	Number of variates m
3	cons	I	1	Input	Criterion value s
4	cp	{D∗ / R∗}	m	Output	Cumulative contribution ratio ck values c_k $(i = 1, 2, \cdots, k_m)$
5	num	I*	1	Output	k_m value
6	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) m ≥ 1

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
3000	Restriction (a) was not satisfied.	Processing is aborted.

(6) **Notes**

(a) The eigenvalues must be arranged in ascending order.

9.2.2 ASL_d6cpsc, ASL_r6cpsc Principal Component Scores

(1) **Function**

Let the mean vector and variance-covariance matrix of the probability vector $\boldsymbol{x} = [x_1, \cdots, x_m]^T$ consisting of m variates be represented by $\boldsymbol{\mu}$ and Σ, respectively. If the eigenvalues of Σ are given by $\lambda_1 \leq \cdots \leq \lambda_m$, and the corresponding orthonormal eigenvector is given by $C = [\boldsymbol{c_1}, \cdots, \boldsymbol{c_m}]$, the principal component score vector of individual j is defined as follows.

$$\boldsymbol{y_l} = C^T(\boldsymbol{x_j} - \boldsymbol{\mu}) \ (j, l = 1, 2, \cdots, m)$$

Given n observed values for m variates x_{ij} $(i = 1, 2, \cdots, n;\ j = 1, 2, \cdots, m)$ and the mean μ_j $(j = 1, 2, \cdots, m)$ of each variate and orthonormal eigenvectors of the variance-covariance matrix $\boldsymbol{c_j} = (c_{jl})$ $(j = 1, 2, \cdots, m; l = 1, 2, \cdots, m)$, the ASL_d6cpsc or ASL_r6cpsc obtains the principal component scores of each individual. Here, $k(k \leq m)$ represents the number of principal components to be obtained. Also, the scores are obtained for standardized variates as follows, where σ_j is the standard deviation of each variate.

$$y_{il} = \sum_{j=1}^{m} \frac{c_{jl}(x_{ij} - \mu_j)}{\sigma_j} \ (i = 1, 2, \cdots, m; l = 1, 2, \cdots, k)$$

(2) **Usage**

Double precision:

```
ierr = ASL_d6cpsc  (a, ma, m, n, num, x1, sd, ev, mev, z);
```

Single precision:

```
ierr = ASL_r6cpsc  (a, ma, m, n, num, x1, sd, ev, mev, z);
```

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	a	{D∗ / R∗}	ma×m	Input	Observation data matrix (x_{ij}) (See Note (a))
2	ma	I	1	Input	Adjustable dimension of array a
3	m	I	1	Input	Number of variates m
4	n	I	1	Input	Number of observed values n
5	num	I	1	Input	Number of principal components k
6	x1	{D∗ / R∗}	m	Input	Mean of each variate μ_j
7	sd	{D∗ / R∗}	m	Input	Standard deviation of each variate σ_j
8	ev	{D∗ / R∗}	mev×m	Input	Matrix composed of eigenvectors (c_{jl}) (See Note (a))
9	mev	I	1	Input	Adjustable dimension of array ev
10	z	{D∗ / R∗}	ma×num	Output	Matrix composed of principal component scores (y_{il}) (See Note (a))
11	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) $2 \leq \mathrm{n} \leq \mathrm{ma}$

(b) $1 \leq \mathrm{m} \leq \mathrm{mev}$

(c) $\mathrm{num} \geq 1$

(d) $\mathrm{sd}[\mathrm{i}-1] \geq$ Unit for determining error$(\mathrm{i} = 1, \cdots, \mathrm{m})$

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
3000	Restriction (a) was not satisfied.	Processing is aborted.
3010	Restriction (b) was not satisfied.	
3020	Restriction (c) was not satisfied.	
3030	Restriction (d) was not satisfied.	

(6) **Notes**

(a) The matrices (x_{ij}) and (c_{jl}) are stored in arrays a and ev, respectively, as real matrices (two-dimensional array type). The matrix (y_{il}) is stored in array z as a real matrix (two-dimensional array type). For the method of the matrix data storage, see Appendix A.

(7) **Example**

(a) Problem
Obtain the eigenvalues and eigenvectors based on the correlation coefficient matrix, and then obtain the cumulative contribution ration of the eigenvalues and the principal component scores.

(b) Input data
Observation data matrix stored in array a:

$$\begin{bmatrix} 90.0 & 91.0 & 98.0 & 90.0 \\ 92.0 & 97.0 & 94.0 & 92.0 \\ 94.0 & 93.0 & 90.0 & 97.0 \\ 97.0 & 94.0 & 95.0 & 95.0 \\ 99.0 & 105.0 & 94.0 & 106.0 \\ 102.0 & 103.0 & 103.0 & 107.0 \\ 104.0 & 95.0 & 110.0 & 110.0 \\ 105.0 & 106.0 & 107.0 & 104.0 \\ 108.0 & 109.0 & 105.0 & 100.0 \\ 109.0 & 107.0 & 104.0 & 99.0 \end{bmatrix}$$

Correlation coefficient matrix stored in array r:

$$\begin{bmatrix} 1.0000 & 0.8000 & 0.7650 & 0.6400 \\ 0.8000 & 1.0000 & 0.4500 & 0.4575 \\ 0.7650 & 0.4500 & 1.0000 & 0.5800 \\ 0.6400 & 0.4575 & 0.5800 & 1.0000 \end{bmatrix}$$

x1[0] = 100.0,
x1[1] = 100.0,
x1[2] = 100.0,
x1[3] = 100.0,
sd[0] = 6.6667,
sd[1] = 6.6667,
sd[2] = 6.6667,
sd[3] = 6.6667,
ma = 10, mev = 4, m = 4, n = 10 and cons = 0.80.

(c) Main program

```
/*      C interface example for ASL_d6cpsc, STAT_d6cpcc */

#include <stdio.h>
#include <stdlib.h>
#include <asl.h>

int main()
{
    double *a;
    int ma;
    int m, n;
    int num;
    double *x1, *sd, *ev;
    int mev;
    double *z, *e;
```

```
double cons;
double *cp;
int ierr;

double *r,*w1;
int i,j;
FILE *fp;

mev= 4;
ma =10;
m  = 4;
n  =10;
cons=0.8;

fp = fopen( "d6cpsc.dat", "r" );
if( fp == NULL )
{
    printf( "file open error\n" );
    return -1;
}

printf( "    *** ASL_d6cpsc ***\n" );
printf( "\n     ** Input **\n\n" );
printf( "\tmev=%3d\n", mev );
printf( "\tma =%3d\n", ma );
printf( "\tm  =%3d\n", m );
printf( "\tn  =%3d\n", n );
printf( "\tcons=%6.3g\n\n", cons );

a = ( double * )malloc((size_t)( sizeof(double) * (ma*m) ));
if( a == NULL )
{
    printf( "no enough memory for array a\n" );
    return -1;
}

x1 = ( double * )malloc((size_t)( sizeof(double) * m ));
if( x1 == NULL )
{
    printf( "no enough memory for array x1\n" );
    return -1;
}

sd = ( double * )malloc((size_t)( sizeof(double) * m ));
if( sd == NULL )
{
    printf( "no enough memory for array sd\n" );
    return -1;
}

r = ( double * )malloc((size_t)( sizeof(double) * (m*m) ));
if( r == NULL )
{
    printf( "no enough memory for array r\n" );
    return -1;
}

ev = ( double * )malloc((size_t)( sizeof(double) * (mev*m) ));
if( ev == NULL )
{
    printf( "no enough memory for array ev\n" );
    return -1;
}

e = ( double * )malloc((size_t)( sizeof(double) * m ));
if( e == NULL )
{
    printf( "no enough memory for array e\n" );
    return -1;
}

z = ( double * )malloc((size_t)( sizeof(double) * (ma*m) ));
if( z == NULL )
{
    printf( "no enough memory for array z\n" );
    return -1;
}

w1 = ( double * )malloc((size_t)( sizeof(double) * m ));
if( w1 == NULL )
{
    printf( "no enough memory for array w1\n" );
    return -1;
}

cp = ( double * )malloc((size_t)( sizeof(double) * m ));
if( cp == NULL )
{
    printf( "no enough memory for array cp\n" );
    return -1;
}

for( j=0 ; j<m ; j++ )
```

```
{
    for( i=0 ; i<n ; i++ )
    {
        fscanf( fp, "%lf", &a[i+ma*j] );
    }
}

printf( "\tArray a\n" );
for( i=0 ; i<n ; i++ )
{
    printf( "\t" );
    for( j=0 ; j<m ; j++ )
    {
        printf( "%8.3g", a[i+ma*j] );
    }
    printf( "\n" );
}
printf( "\n" );

for( j=0 ; j<m ; j++ )
{
    for( i=0 ; i<m ; i++ )
    {
        fscanf( fp, "%lf", &r[i+m*j] );
    }
}

printf( "\tArray r\n" );
for( i=0 ; i<m ; i++ )
{
    printf( "\t" );
    for( j=0 ; j<m ; j++ )
    {
        printf( "%8.3g", r[i+m*j] );
    }
    printf( "\n" );
}
printf( "\n" );

printf( "\tArray x1\n" );
printf( "\t" );
for( i=0 ; i<m ; i++ )
{
    fscanf( fp, "%lf", &x1[i] );
    printf( "%8.3g", x1[i] );
}
printf( "\n\n" );

printf( "\tArray sd\n" );
printf( "\t" );
for( i=0 ; i<m ; i++ )
{
    fscanf( fp, "%lf", &sd[i] );
    printf( "%8.3g", sd[i] );
}
printf( "\n" );

fclose( fp );

for( j=0 ; j<m ; j++ )
{
    for( i=0 ; i<m ; i++ )
    {
        ev[i+mev*j]=r[i+m*j];
    }
}

ierr = ASL_dcsmaa(ev, mev, m, e, w1);

printf( "\n     ** Output (ASL_dcsmaa) **\n\n" );
printf( "\tierr = %6d\n\n", ierr );

printf( "\tArray e (Eigenvalues)\n" );
printf( "\t" );
for( i=0 ; i<m ; i++ )
{
    printf( "%8.3g", e[i] );
}
printf( "\n" );
printf( "\tArray ev (Eigenvectors)\n" );
for( i=0 ; i<m ; i++ )
{
    printf( "\t" );
    for( j=0 ; j<m ; j++ )
    {
        printf( "%8.3g", ev[i+mev*j] );
    }
    printf( "\n" );
```

```
    }

    ierr = ASL_d6cpcc(e, m, cons, cp, &num);

    printf( "\n     ** Output (ASL_d6cpcc) **\n\n" );
    printf( "\tierr = %6d\n", ierr );
    printf( "\tnum  = %6d\n\n", num );

    printf( "\tArray cp (Cumulative Ratio)\n" );
    printf( "\t" );
    for( i=0 ; i<num ; i++ )
    {
        printf( "%8.3g", cp[i] );
    }
    printf( "\n" );

    ierr = ASL_d6cpsc(a, ma, m, n, num, x1, sd, ev, mev, z);

    printf( "\n     ** Output (ASL_d6cpsc) **\n\n" );
    printf( "\tierr = %6d\n\n", ierr );

    printf( "\tArray z (Principal Component Score)\n" );
    for( i=0 ; i<n ; i++ )
    {
        printf( "\t" );
        for( j=0 ; j<num ; j++ )
        {
            printf( "%8.3g\t", z[i+ma*j] );
        }
        printf( "\n" );
    }

    free( a );
    free( x1 );
    free( sd );
    free( r );
    free( ev );
    free( e );
    free( z );
    free( w1 );
    free( cp );

    return 0;
}
```

(d) Output results

```
*** ASL_d6cpsc ***

 ** Input **

mev=  4
ma = 10
m  =  4
n  = 10
cons=   0.8

Array a
      90      91      98      90
      92      97      94      92
      94      93      90      97
      97      94      95      95
      99     105      94     106
     102     103     103     107
     104      95     110     110
     105     106     107     104
     108     109     105     100
     109     107     104      99

Array r
       1     0.8   0.765    0.64
     0.8       1    0.45   0.458
   0.765    0.45       1    0.58
    0.64   0.458    0.58       1

Array x1
     100     100     100     100

Array sd
    6.67    6.67    6.67    6.67

 ** Output (ASL_dcsmaa) **

ierr =      0

Array e (Eigenvalues)
  0.0923   0.435    0.61    2.86
Array ev (Eigenvectors)
   0.791   0.157  -0.175   0.565
  -0.468  -0.161  -0.728   0.475
  -0.387   0.656   0.424   0.491
 -0.0747  -0.721    0.51   0.463
```

```
 ** Output (ASL_d6cpcc) **

ierr =      0
num  =      2

Array cp (Cumulative Ratio)
   0.716   0.868

 ** Output (ASL_d6cpsc) **

ierr =      0

Array z (Principal Component Score)
   -2.33       0.353
   -1.89      -0.456
   -1.95      0.0566
    -1.4      0.0334
   0.247      -0.442
    1.09       0.346
    1.41        1.84
    1.64     -0.0353
    1.69      -0.875
    1.49      -0.823
```

9.3 FACTOR ANALYSIS

9.3.1 ASL_d6fald, ASL_r6fald
Factor Loading Matrix

(1) **Function**

The ASL_d6fald or ASL_r6fald obtains the factor loading matrix and communality (contribution ratio of principal components) based on the eigenvalues and eigenvectors.

Factor loading matrix (a_{ij}) :

$$a_{ij} = \sqrt{\lambda_j} v_{ij} (i = 1, 2, \cdots, m \text{ (Number of variates)};\ j = 1, 2, \cdots, k \text{ (Number of factors)};\ k \leq m)$$

Here, λ_j $(\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_k \geq 0)$ is eigenvalues and v_{ij} is the ith component of the eigenvector for eigenvalue λ_j.

Communality

$$h_i^2 = \sum_{j=1}^{k} a_{ij}^2$$

(2) **Usage**

Double precision:

```
ierr = ASL_d6fald  (e, m, ev, lme, num, fm, lmf, oc);
```

Single precision:

```
ierr = ASL_r6fald  (e, m, ev, lme, num, fm, lmf, oc);
```

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	e	{D∗ / R∗}	m	Input	Eigenvalues λ_j (See Notes (a) and (b))
2	m	I	1	Input	Number of eigenvalues m
3	ev	{D∗ / R∗}	lme×m	Input	Matrix composed of eigenvectors corresponding to each eigenvalue (v_{ij}) (See Note (a))
4	lme	I	1	Input	Adjustable dimension of array ev
5	num	I	1	Input	Number of factors k
6	fm	{D∗ / R∗}	lmf×num	Output	Factor loading matrix (a_{ij})
7	lmf	I	1	Input	Adjustable dimension of array fm

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
8	oc	$\left\{\begin{matrix}\text{D}* \\ \text{R}*\end{matrix}\right\}$	m	Output	Initial communality (contribution ratio for variates) h_i^2
9	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) 1≤num≤m≤lme, lmf

(b) e[i-1](i=1, $\cdots$ m) must be arranged in ascending order.

(c) e[m−num]≥0

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
3000	Restriction (a) was not satisfied.	Processing is aborted.
3010	Restriction (b) was not satisfied.	
3020	Restriction (c) was not satisfied.	

(6) **Notes**

(a) The eigenvalues must be arranged in ascending order. Also, the eigenvectors must be arranged corresponding to the eigenvalues. Matrix composed of eigenvectors (v_{ij}) and factor loading matrix (a_{ij}) are stored in arrays ev and fm as real matrices (two-dimensional array type) (See Appendix A).

(b) The largest num eigenvalues and corresponding eigenvectors are used for the calculation.

9.3.2 ASL_d6favr, ASL_r6favr Rotation According to the Varimax Criterion

(1) **Function**

The ASL_d6favr or ASL_r6favr performs orthogonal rotations of the factor loading matrix according to the varimax criterion

Also obtain the following values.

Communality

$$h_i^2 = \sum_{j=1}^{k} a_{ij}^2 \quad (i = 1, 2, \cdots, m \text{ (Number of variates)};\ j = 1, 2, \cdots, k \text{ (Number of factors)})$$

Here, $A = (a_{ij})$ is the factor loading matrix.

Variance of factor loading matrix

$$V_c = \sum_{j=1}^{k} \frac{m \sum_{i=1}^{m} (b_{ij}^2)^2 - (\sum_{i=1}^{m} b_{ij}^2)^2}{m^2} \quad (c = 1, 2, \cdots, r(\text{Maximum number of orthogonal rotations}))$$

Here, b_{ij} is defined as follows.

$$b_{ij} = a_{ij} / \sqrt{h_i^2} \quad (i = 1, 2, \cdots, m; \quad j = 1, 2, \cdots, k)$$

(2) **Usage**

Double precision:

```
ierr = ASL_d6favr  (fm, lmf, m, num, &ic, com, lmc, v);
```

Single precision:

```
ierr = ASL_r6favr  (fm, lmf, m, num, &ic, com, lmc, v);
```

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	fm	{D∗ / R∗}	lmf×num	Input	Factor loading matrix before rotations (See Note (a))
				Output	Factor loading matrix after final rotation
2	lmf	I	1	Input	Adjustable dimension of array fm
3	m	I	1	Input	Number of variates m
4	num	I	1	Input	Number of factors k
5	ic	I*	1	Input	Maximum number of orthogonal rotations (See Note (b))
				Output	Actual number of orthogonal rotations
6	com	{D∗ / R∗}	lmc× 3	Output	com[i-1]: Communality before rotations com[i-1+lmc]: Communality after final rotation com[i-1+2× lmc]: (Communality before rotations) − (Communality after final rotation) (i = 1, ··· , m)
7	lmc	I	1	Input	Adjustable dimension of array com
8	v	{D∗ / R∗}	ic+1	Output	Variance of factor matrix for each rotation v[0]: Variance before rotations v[i]: Variance after i-th rotation (i=1, ···, ic)
9	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) 1≤num≤m≤lmf, lmc

(b) ic ≥ 1

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
3000	Restriction (a) was not satisfied.	Processing is aborted.
3010	Restriction (b) was not satisfied.	
5000	The solution did not converge even though the assigned maximum number of orthogonal rotations was reached.	The factor loading matrix, communality, and variance at that time are returned.

(6) **Notes**

(a) Factor loading matrix (a_{ij}) are stored in array fm as real matrix (two-dimensional array type) (See Appendix A).

(b) A value of approximately 50 is suitable for ic.

(7) **Example**

(a) Problem

Obtain the eigenvalues, eigenvectors, and factor loading matrix of the following correlation coefficient matrix:

$$A = \begin{bmatrix} 1.00000 & 0.80000 & 0.76500 & 0.64000 \\ 0.80000 & 1.00000 & 0.45000 & 0.45750 \\ 0.76500 & 0.45000 & 1.00000 & 0.58000 \\ 0.64000 & 0.45750 & 0.58000 & 1.00000 \end{bmatrix}$$

and performs orthogonal rotations of the factor loading matrix according to the varimax criterion and obtain the factor loading matrix after rotations and so on.

(b) Input data

Correlation coefficient matrix A, m=4, lme=5, num=2, lmf=5, ic=10 and lmc=5.

(c) Main program

```
/*      C interface example for ASL_d6favr */

#include <stdio.h>
#include <stdlib.h>
#include <asl.h>

int main()
{
    double *fm;
    int lmf;
    int m;
    int num;
    int ic;
    double *com;
    int lmc;
    double *v;
    int ierr;

    double *a, *ev, *oc, *w1;
    int lme;
    int maxic;
    int i,j;

    FILE *fp;


    m=4;
    lme=5;
    num=2;
    lmf=5;
    maxic=10;
    lmc=5;
    ic=maxic;

    fp = fopen( "d6favr.dat", "r" );
    if( fp == NULL )
    {
        printf( "file open error\n" );
        return -1;
    }

    printf( "    *** ASL_d6favr ***\n" );
    printf( "\n     ** Input **\n\n" );
    printf( "\t  m=%3d\n", m );
    printf( "\tlme=%3d\n", lme );
    printf( "\tnum=%3d\n", num );
    printf( "\tlmf=%3d\n", lmf );
    printf( "\t ic=%3d\n", ic );
    printf( "\tlmc=%3d\n\n", lmc );

    a = ( double * )malloc((size_t)( sizeof(double) * (m) ));
    if( a == NULL )
```

```
{
    printf( "no enough memory for array a\n" );
    return -1;
}

ev = ( double * )malloc((size_t)( sizeof(double) * (lme*m) ));
if( ev == NULL )
{
    printf( "no enough memory for array ev\n" );
    return -1;
}

fm = ( double * )malloc((size_t)( sizeof(double) * (lmf*num) ));
if( fm == NULL )
{
    printf( "no enough memory for array fm\n" );
    return -1;
}

oc = ( double * )malloc((size_t)( sizeof(double) * (m) ));
if( oc == NULL )
{
    printf( "no enough memory for array oc\n" );
    return -1;
}

w1 = ( double * )malloc((size_t)( sizeof(double) * (m) ));
if( w1 == NULL )
{
    printf( "no enough memory for array w1\n" );
    return -1;
}

com = ( double * )malloc((size_t)( sizeof(double) * (lmc*3) ));
if( com == NULL )
{
    printf( "no enough memory for array com\n" );
    return -1;
}

v = ( double * )malloc((size_t)( sizeof(double) * (maxic+1) ));
if( v == NULL )
{
    printf( "no enough memory for array v\n" );
    return -1;
}

printf( "\tCorrelation matrix\n\t" );

for( i=0 ; i<m ; i++ )
{
    for( j=0 ; j<m ; j++ )
    {
        fscanf( fp, "%lf", &ev[i+lme*j] );
        printf( "%8.3g ", ev[i+lme*j] );
    }
    printf( "\n\t" );
}

fclose( fp );

printf( "\n     ** Output **\n\n" );

ierr = ASL_dcsmaa(ev, lme, m, a, w1);

printf( "\t** ASL_dcsmaa **\n\n" );
printf( "\tierr = %6d\n\n", ierr );
printf( "\tEigen value\n" );
printf( "\t" );
for( i=0 ; i<m ; i++ )
{
    printf( "%8.3g ", a[i] );
}
printf( "\n" );
printf( "\tEigen vector\n" );
for( i=0 ; i<m ; i++ )
{
    printf( "\t" );
    for( j=0 ; j<m ; j++ )
    {
        printf( "%8.3g ", ev[i+lme*j] );
    }
    printf( "\n" );
}
printf( "\n" );

ierr = ASL_d6fald(a, m, ev, lme, num, fm, lmf, oc);

printf( "\t** ASL_d6fald **\n\n" );
printf( "\tierr = %6d\n\n", ierr );
printf( "\tFactor loading matrix\n" );
```

```
    for( j=0 ; j<num ; j++ )
    {
        printf( "\t" );
        for( i=0 ; i<m ; i++ )
        {
            printf( "%8.3g ", fm[i+lmf*j] );
        }
        printf( "\n" );
    }
    printf( "\n" );

    printf( "\tCommunalities\n" );
    printf( "\t" );
    for( i=0 ; i<m ; i++ )
    {
        printf( "%8.3g ", oc[i] );
    }
    printf( "\n\n" );

    ierr = ASL_d6favr(fm, lmf, m, num, &ic, com, lmc, v);

    printf( "\t** ASL_d6favr **\n\n" );
    printf( "\tierr = %6d\n\n", ierr );

    printf( "\tFactor loading matrix\n" );
    for( j=0 ; j<num ; j++ )
    {
        printf( "\t" );
        for( i=0 ; i<m ; i++ )
        {
            printf( "%8.3g ", fm[i+lmf*j] );
        }
        printf( "\n" );
    }
    printf( "\n" );

    printf( "\tic = %6d\n\n", ic );

    for( i=0 ; i<ic+1 ; i++ )
    {
        printf( "\tv[%2d] = %8.3g\n", i,v[i] );
    }
    printf( "\n" );

    printf( "\tCommunalities\n" );
    printf( "\t" );
    for( i=0 ; i<m ; i++ )
    {
        printf( "%8.3g ", oc[i] );
    }
    printf( "\n\n" );

    printf( "\tvariable  original     final  difference\n" );
    for( i=0 ; i<m ; i++ )
    {
        printf( "\t      %2d  %8.3g  %8.3g  %8.3g\n",
            i, com[i], com[i+lmc], com[i+lmc*2] );
    }

    free( a );
    free( ev );
    free( fm );
    free( oc );
    free( w1 );
    free( com );
    free( v );

    return 0;
}
```

(d) Output results

```
*** ASL_d6favr ***

 ** Input **

  m=  4
lme=  5
num=  2
lmf=  5
 ic= 10
lmc=  5

Correlation matrix
       1      0.8    0.765     0.64
     0.8        1     0.45    0.458
   0.765     0.45        1     0.58
    0.64    0.458     0.58        1

 ** Output **
```

```
** ASL_dcsmaa **

ierr =      0

Eigen value
   0.0923    0.435     0.61     2.86
Eigen vector
    0.791    0.157   -0.175    0.565
   -0.468   -0.161   -0.728    0.475
   -0.387    0.656    0.424    0.491
  -0.0747   -0.721     0.51    0.463

** ASL_d6fald **

ierr =      0

Factor loading matrix
    0.955    0.803    0.831    0.784
   -0.136   -0.568    0.331    0.398

Communalities
    0.931    0.968    0.799    0.773

** ASL_d6favr **

ierr =      0

Factor loading matrix
    0.622    0.221     0.84     0.85
   -0.738   -0.959   -0.306   -0.225

ic =      4

v[ 0] =   0.0257
v[ 1] =    0.195
v[ 2] =    0.237
v[ 3] =    0.262
v[ 4] =    0.262

Communalities
    0.931    0.968    0.799    0.773

variable  original     final  difference
       0     0.931     0.931         0
       1     0.968     0.968  -1.11e-16
       2     0.799     0.799  -2.22e-16
       3     0.773     0.773   1.11e-16
```

9.4 CANONICAL CORRELATION ANALYSIS

9.4.1 ASL_d6cvan, ASL_r6cvan Canonical Correlation Analysis

(1) **Function**

To perform a canonical correlation analysis for two groups of observed values, the ASL_d6cvan or ASL_r6cvan performs the following processing.
Given the correlation coefficient matrix of the first group R_{11} (size: $m_1 \times m_1$),
the correlation coefficient matrix of the second group R_{22} (size: $m_2 \times m_2$),
and the correlation coefficient matrix of the first and second groups R_{12} (size: $m_1 \times m_2$),
solve the following eigenvalue problem to perform a correlation analysis.

$$
\begin{aligned}
{R_{11}}^{-1} R_{12} {R_{22}}^{-1} {R_{12}}^{T} \boldsymbol{p} &= \lambda^2 \boldsymbol{p} \\
\boldsymbol{q} &= \lambda^{-1} {R_{22}}^{-1} {R_{12}}^{T} \boldsymbol{p}
\end{aligned}
$$

Define the following kind of matrix R by collecting together the correlation coefficient matrices.

$$
\begin{bmatrix}
R_{11} & {R_{12}}^{T} \\
R_{12} & R_{22}
\end{bmatrix}
$$

Next, from the eigenvalues ${\lambda_i}^2$ and eigenvectors $\boldsymbol{p_i}$ and $\boldsymbol{q_i}$ that were found, obtain the canonical correlation coefficients, Wilks' Λ, as well as the canonical coefficients of each group, which are defined by the following equations.
Canonical correlation coefficients=λ_i $(i = 1, \cdots, m)$
However, $\lambda_1 > \lambda_2 > \cdots > \lambda_m$, $m = min(m_1, m_2)$.
Wilks' Λ:

$$
\Lambda_k = \prod_{i=k+1}^{l} (1 - {\lambda_i}^2)
$$

Canonical coefficients of first group: $(\boldsymbol{p_1}, \boldsymbol{p_2}, \cdots, \boldsymbol{p_m})$
Canonical coefficients of second group: $(\boldsymbol{q_1}, \boldsymbol{q_2}, \cdots, \boldsymbol{q_m})$
The number of nonzero canonical correlation coefficients is said to be the number of dimensions of a canonical variate. The number of dimensions can be determined by sequentially performing hypothesis tests that test the following null hypothesis:

$$
H_k : \lambda_{k+1} = \cdots = \lambda_l = 0
$$

against the following alternative hypothesis:

$$
K_k : H_k \text{ is not true}
$$

That is, if $H_0, \cdots, H_{k-1}$ are rejected and H_k is adopted, the number of dimensions is assumed to be k. The test is performed by using the following fact. If Wilks' Λ is used, based on hypothesis H_k, the following χ_k^2 values:

$$
\chi_k^2 = -\{n - 0.5(m_1 + m_2 + 1)\} \log_e \Lambda_k
$$

asymptotically obey a χ^2 distribution with $(m_1 - k)(m_2 - k)$ degrees of freedom.

(2) **Usage**

Double precision:

ierr = ASL_d6cvan (n, m1, m2, r, mr, co, co1, mco1, co2, mco2, e, wil, chi, ndf, w1);

Single precision:

ierr = ASL_r6cvan (n, m1, m2, r, mr, co, co1, mco1, co2, mco2, e, wil, chi, ndf, w1);

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	n	I	1	Input	Number of observed values
2	m1	I	1	Input	Number of variates of first group
3	m2	I	1	Input	Number of variates of second group
4	r	{D∗ / R∗}	See Contents	Input	Correlation coefficient matrix R **Size**: mr×(m1 + m2)
5	mr	I	1	Input	Adjustable dimension of array r
6	co	{D∗ / R∗}	m1	Output	Canonical correlation coefficients
7	co1	{D∗ / R∗}	mco1×m1	Output	Canonical coefficient matrix of first group
8	mco1	I	1	Input	Adjustable dimension of array co1
9	co2	{D∗ / R∗}	mco2×m1	Output	Canonical coefficient matrix of second group
10	mco2	I	1	Input	Adjustable dimension of array co2
11	e	{D∗ / R∗}	m1	Output	Eigenvalues
12	wil	{D∗ / R∗}	m1	Output	Wilks' Λ
13	chi	{D∗ / R∗}	m1	Output	χ^2 values χ_k^2
14	ndf	I*	m1	Output	Number of degrees of freedom for χ^2 test
15	w1	{D∗ / R∗}	See Contents	Work	Work area **Size**: mr × (m1 + m2)
16	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) n≥2

(b) m1≤m2

(c) m1+m2≤mr

(d) 2≤m1≤mco1, 2≤m2≤mco2

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
3000	Restriction (a) was not satisfied.	Processing is aborted.
3010	Restriction (b) was not satisfied.	
3020	Restriction (c) was not satisfied.	
3030	Restriction (d) was not satisfied.	
4000+i	When performing an LU decomposition, the pivot was 0.0 during the processing of the i-th step.	
5000	The solution did not converge during the step for obtaining the eigenvalues.	Processing is aborted.
6000	The eigenvalue was smaller than the unit for determining error.	

(6) **Notes**

None

9.4.2 ASL_d6cvsc, ASL_r6cvsc Canonical Variate Scores

(1) **Function**

The ASL_d6cvsc or ASL_r6cvsc obtains the canonical variate values (scores) of the observed values based on the canonical coefficients (matrix). The canonical variate z_i of the observed values $x_{i,j}$ $(i = 1, 2, \cdots, n;\ j = 1, 2, \cdots, m_1)$ for m_1 variables x_j $(j = 1, 2, \cdots, m_1)$ concerning n subjects is defined by the following equation.

$$z_i = \sum_{j=1}^{m} p_i \frac{x_{i,j} - \overline{x_j}}{\sigma_j}$$

Here, $\overline{x_j}$ and σ_j^2 represent the mean and variance of the observed values and $\boldsymbol{p} = \{p_i\}$ represents the canonical coefficient vector.

(2) **Usage**

Double precision:

ierr = ASL_d6cvsc (n, m1, m2, a, ma, co1, mco1, co2, mco2, x1, sd, z);

Single precision:

ierr = ASL_r6cvsc (n, m1, m2, a, ma, co1, mco1, co2, mco2, x1, sd, z);

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	n	I	1	Input	Number of observed values
2	m1	I	1	Input	Number of variates of first group
3	m2	I	1	Input	Number of variates of second group
4	a	{D∗ / R∗}	See Contents	Input	Observed value matrix **Size**: ma×(m1 + m2)
5	ma	I	1	Input	Adjustable dimension of arrays a and z
6	co1	{D∗ / R∗}	mco1×m1	Input	Canonical coefficient matrix of first group
7	mco1	I	1	Input	Adjustable dimension of array co1
8	co2	{D∗ / R∗}	mco2×m1	Input	Canonical coefficient matrix of second group
9	mco2	I	1	Input	Adjustable dimension of array co2
10	x1	{D∗ / R∗}	m1+m2	Input	Mean of each variate
11	sd	{D∗ / R∗}	m1+m2	Input	Standard deviation of each variate
12	z	{D∗ / R∗}	See Contents	Output	Canonical variate values **Size**: ma×(2 × m1)
13	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) 2≤n≤ma

(b) 2≤m1≤mco1, 2≤m2≤mco2

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
3000	Restriction (a) was not satisfied.	Processing is aborted.
3010	Restriction (b) was not satisfied.	

(6) **Notes**

(a) The canonical variate values are stored in array z as the following kind of real matrix (two-dimensional array type) (See Appendix A).

$$\begin{bmatrix} u_{1,1} & u_{1,2} & \cdots & u_{1,\mathrm{m1}} & v_{1,1} & v_{1,2} & \cdots & v_{1,\mathrm{m1}} \\ u_{2,1} & \ddots & & \vdots & v_{2,1} & \ddots & & \vdots \\ \vdots & & \ddots & \vdots & \vdots & & \ddots & \vdots \\ u_{\mathrm{n},1} & \cdots & \cdots & u_{\mathrm{n},\mathrm{m1}} & v_{\mathrm{n},1} & \cdots & \cdots & v_{\mathrm{n},\mathrm{m1}} \end{bmatrix}$$

Here, $u_{k,i}$ and $v_{k,j}$ have the following meanings.
$u_{k,i}$ $(i = 1, 2, \cdots, \mathrm{m1}; k = 1, 2, \cdots, \mathrm{n})$: Canonical variate of first group for i-th canonical correlation coefficient
$v_{k,j}$ $(j = 1, 2, \cdots, \mathrm{m1}; k = 1, 2, \cdots, \mathrm{n})$: Canonical variate of second group for j-th canonical correlation coefficient

(7) **Example**

(a) Problem
Obtain the canonical coefficient matrix from the following observed value matrix.

$$A = \begin{bmatrix} 90.0 & 91.0 & 95.0 & 103.0 & 75.0 \\ 97.0 & 98.0 & 98.0 & 92.0 & 76.0 \\ 93.0 & 92.0 & 97.0 & 106.0 & 77.0 \\ 99.0 & 90.0 & 99.0 & 108.0 & 78.0 \\ 102.0 & 97.0 & 101.0 & 105.0 & 79.0 \\ 100.0 & 100.0 & 100.0 & 100.0 & 80.0 \\ 103.0 & 99.0 & 103.0 & 103.0 & 81.0 \\ 106.0 & 98.0 & 101.0 & 101.0 & 82.0 \\ 111.0 & 90.0 & 99.0 & 104.0 & 83.0 \\ 108.0 & 98.0 & 103.0 & 102.0 & 84.0 \\ 104.0 & 97.0 & 105.0 & 99.0 & 85.0 \end{bmatrix}$$

Also obtain the canonical correlation coefficients and canonical variates.

(b) Input data
Observed value matrix A, n=11, m1=2, m2=3, mr=5, ma=11, mco1=3 and mco2=3.

(c) Main program

```
/*      C interface example for ASL_d6cvsc, STAT_d6cvan */

#include <stdio.h>
#include <stdlib.h>
#include <asl.h>

int main()
{
    int n, m1, m2;
    double *a;
    int ma;
    double *co1, *co2;
    int mco1, mco2;
    double *x1, *stat, *sd, *z;

    double *r;
    int mr;
    double *co, *e, *wil, *chi;
    int *ndf;
    double *w1;
    int ierr;

    int m,isw;
    int i,j;
```

```
int ns;
FILE *fp;

n=11;
m1=2;
m2=3;
mr=5;
mco1=3;
mco2=3;
ma=11;
m=m1+m2;

fp = fopen( "d6cvsc.dat", "r" );
if( fp == NULL )
{
    printf( "file open error\n" );
    return -1;
}

printf( "    *** ASL_d6cvsc ***\n" );
printf( "\n     ** Input **\n\n" );
printf( "\tn   =%3d\n", n );
printf( "\tm1  =%3d\n", m1 );
printf( "\tm2  =%3d\n", m2 );
printf( "\tmr  =%3d\n", mr );
printf( "\tmco1=%3d\n", mco1 );
printf( "\tmco2=%3d\n", mco2 );
printf( "\tma  =%3d\n\n", ma );

a = ( double * )malloc((size_t)( sizeof(double) * (ma*m) ));
if( a == NULL )
{
    printf( "no enough memory for array a\n" );
    return -1;
}

r = ( double * )malloc((size_t)( sizeof(double) * (mr*m) ));
if( r == NULL )
{
    printf( "no enough memory for array r\n" );
    return -1;
}

co = ( double * )malloc((size_t)( sizeof(double) * m1 ));
if( co == NULL )
{
    printf( "no enough memory for array co\n" );
    return -1;
}

e = ( double * )malloc((size_t)( sizeof(double) * m1 ));
if( e == NULL )
{
    printf( "no enough memory for array e\n" );
    return -1;
}

wil = ( double * )malloc((size_t)( sizeof(double) * m1 ));
if( wil == NULL )
{
    printf( "no enough memory for array wil\n" );
    return -1;
}

chi = ( double * )malloc((size_t)( sizeof(double) * m1 ));
if( chi == NULL )
{
    printf( "no enough memory for array chi\n" );
    return -1;
}

ndf = ( int * )malloc((size_t)( sizeof(int) * m1 ));
if( ndf == NULL )
{
    printf( "no enough memory for array ndf\n" );
    return -1;
}

co1 = ( double * )malloc((size_t)( sizeof(double) * (mco1*m1) ));
if( co1 == NULL )
{
    printf( "no enough memory for array co1\n" );
    return -1;
}

co2 = ( double * )malloc((size_t)( sizeof(double) * (mco2*m1) ));
if( co2 == NULL )
{
    printf( "no enough memory for array co2\n" );
    return -1;
}

w1 = ( double * )malloc((size_t)( sizeof(double) * (mr*m) ));
```

```
if( w1 == NULL )
{
    printf( "no enough memory for array w1\n" );
    return -1;
}

x1 = ( double * )malloc((size_t)( sizeof(double) * m ));
if( x1 == NULL )
{
    printf( "no enough memory for array x1\n" );
    return -1;
}

sd = ( double * )malloc((size_t)( sizeof(double) * m ));
if( sd == NULL )
{
    printf( "no enough memory for array sd\n" );
    return -1;
}

z = ( double * )malloc((size_t)( sizeof(double) * (ma*2*m1) ));
if( z == NULL )
{
    printf( "no enough memory for array z\n" );
    return -1;
}

stat = ( double * )malloc((size_t)( sizeof(double) * (m*5) ));
if( stat == NULL )
{
    printf( "no enough memory for array stat\n" );
    return -1;
}

for( j=0 ; j<m ; j++ )
{
    for( i=0 ; i<n ; i++ )
    {
        fscanf( fp, "%lf", &a[i+ma*j] );
    }
}
printf( "\tArray a\n" );
for( i=0 ; i<n ; i++ )
{
    printf( "\t" );
    for( j=0 ; j<m ; j++ )
    {
        printf( "%8.3g", a[i+ma*j] );
    }
    printf( "\n" );
}
fclose( fp );

isw=0;

ierr = ASL_d2bams(a, ma, n, m, &ns, stat, isw);

printf( "\n     ** Output (ASL_d2bams) **\n\n" );
printf( "\tierr = %6d\n", ierr );

for( i=0 ; i<m ; i++ )
{
    x1[i]=stat[i+m];
    sd[i]=stat[i+m*4];
}

ierr = ASL_d2ccmt(a, ma, n, m, &ns, x1, r, mr, isw, w1);

printf( "\n     ** Output (ASL_d2ccmt) **\n\n" );
printf( "\tierr = %6d\n\n", ierr );

printf( "\tArray x1(Mean of variables)\n" );
printf( "\t" );
for( i=0 ; i<m ; i++ )
{
    printf( "%8.3g", x1[i] );
}
printf( "\n\n" );

printf( "\tArray sd(Standard deviation)\n" );
printf( "\t" );
for( i=0 ; i<m ; i++ )
{
    printf( "%8.3g", sd[i] );
}
printf( "\n\n" );

printf( "\tArray r(Correlation matrix)\n" );
for( i=0 ; i<m ; i++ )
{
    printf( "\t" );
```

```
        for( j=0 ; j<m ; j++ )
        {
            printf( " %7.3f", r[i+mr*j] );
        }
        printf( "\n" );
    }

    ierr = ASL_d6cvan(n, m1, m2, r, mr, co,
            co1, mco1, co2, mco2, e, wil, chi, ndf, w1);

    printf( "\n     ** Output (ASL_d6cvan) **\n\n" );
    printf( "\tierr = %6d\n\n", ierr );

    printf( "\t      co        e      wil      chi    ndf\n" );
    for( i=0 ; i<m1 ; i++ )
    {
        printf( "\t%8.3f %8.3f %8.3f %8.1f %6d\n",
            co[i], e[i], wil[i], chi[i], ndf[i] );
    }
    printf( "\n" );

    printf( "\tArray co1(Canonical coefficient of the first set)\n" );
    for( j=0 ; j<m1 ; j++ )
    {
        printf( "\t" );
        for( i=0 ; i<m1 ; i++ )
        {
            printf( "%8.3g", co1[i+mco1*j] );
        }
        printf( "\n" );
    }
    printf( "\n" );

    printf( "\tArray co2(Canonical coefficient of the second set)\n" );
    for( j=0 ; j<m1 ; j++ )
    {
        printf( "\t" );
        for( i=0 ; i<m2 ; i++ )
        {
            printf( "%8.3g", co2[i+mco2*j] );
        }
        printf( "\n" );
    }

    ierr = ASL_d6cvsc(n, m1, m2, a, ma, co1, mco1, co2, mco2, x1, sd, z);

    printf( "\n     ** Output (ASL_d6cvsc) **\n\n" );
    printf( "\tierr = %6d\n\n", ierr );

    printf( "\tArray z(Canonical score of the first set)\n" );
    for( i=0 ; i<n ; i++ )
    {
        printf( "\t" );
        for( j=0 ; j<m1 ; j++ )
        {
            printf( "%8.3g", z[i+ma*j] );
        }
        printf( "\n" );
    }
    printf( "\n" );

    printf( "\tArray z(Canonical score of the second set)\n" );
    for( i=0 ; i<n ; i++ )
    {
        printf( "\t" );
        for( j=m1 ; j<2*m1 ; j++ )
        {
            printf( "%8.3g", z[i+ma*j] );
        }
        printf( "\n" );
    }

    free( a );
    free( r );
    free( co );
    free( e );
    free( wil );
    free( chi );
    free( ndf );
    free( co1 );
    free( co2 );
    free( w1 );
    free( x1 );
    free( sd );
    free( z );
    free( stat );

    return 0;
}
```

(d) Output results

```
*** ASL_d6cvsc ***

 ** Input **

n   = 11
m1  =  2
m2  =  3
mr  =  5
mco1=  3
mco2=  3
ma  = 11

Array a
      90      91      95     103      75
      97      98      98      92      76
      93      92      97     106      77
      99      90      99     108      78
     102      97     101     105      79
     100     100     100     100      80
     103      99     103     103      81
     106      98     101     101      82
     111      90      99     104      83
     108      98     103     102      84
     104      97     105      99      85

 ** Output (ASL_d2bams) **

ierr =      0

 ** Output (ASL_d2ccmt) **

ierr =      0

Array x1(Mean of variables)
     101    95.5     100     102      80

Array sd(Standard deviation)
    6.27    3.86    2.91    4.25    3.32

Array r(Correlation matrix)
   1.000   0.257   0.694  -0.008   0.879
   0.257   1.000   0.610  -0.582   0.328
   0.694   0.610   1.000  -0.122   0.848
  -0.008  -0.582  -0.122   1.000  -0.014
   0.879   0.328   0.848  -0.014   1.000

 ** Output (ASL_d6cvan) **

ierr =      0

      co        e      wil      chi    ndf
   0.891    0.794    0.067     21.6      6
   0.821    0.675    0.325      9.0      2

Array co1(Canonical coefficient of the first set)
   0.874   0.312
  -0.554   0.987

Array co2(Canonical coefficient of the second set)
   0.159   -0.18   0.839
    1.33   -0.55   -1.34

 ** Output (ASL_d6cvsc) **

ierr =      0

Array z(Canonical score of the first set)
   -1.92  -0.152
  -0.377    1.02
   -1.42  -0.161
  -0.744    -1.2
   0.239   0.323
   0.203    1.27
    0.54   0.746
   0.877   0.226
   0.927   -2.26
    1.16  0.0488
   0.517   0.146

Array z(Canonical score of the second set)
   -1.58  -0.429
  -0.699    1.96
   -1.09  -0.709
  -0.816  -0.457
  -0.327   0.442
  0.0836   0.229
   0.373   0.809
   0.602   -0.25
   0.619   -1.95
    1.17  -0.271
    1.66   0.628
```

9.5 DISCRIMINANT ANALYSIS

9.5.1 ASL_d6dafn, ASL_r6dafn Discriminant Functions

(1) **Function**

Given g groups of observed values ${x_{l,i}}^{(k)}$ $(l = 1, 2, \cdots, n_k; i = 1, 2, \cdots, m; k = 1, 2, \cdots, g)$ for which $\boldsymbol{\nu_k} = (\overline{x}_{\cdot i}^{(k)})$ where ${\overline{x}_{\cdot i}}^{(k)}$ is defined as follows:

$$ {\overline{x}_{\cdot i}}^{(k)} = \frac{1}{n_k} \sum_{l=1}^{n_k} x_{l,i}^{(k)} $$

is the mean vector of the variates of the k-th group and $\Sigma = (\sigma_{i,j})$ where $\sigma_{i,j}$ is defined as follows:

$$ \sigma_{i,j} = \frac{\sum_{k=1}^{g} \sum_{l=1}^{n_k} (x_{l,i}^{(k)} - {\overline{x}_{\cdot i}}^{(k)})(x_{l,j}^{(k)} - {\overline{x}_{\cdot j}}^{(k)})}{\sum_{k=1}^{g} (n_k - 1)} $$

is the variance-covariance matrix over all groups and for which the populations for the observed values are normal populations of order m represented by $N(\boldsymbol{\nu_1}, \Sigma), \cdots, N(\boldsymbol{\nu_k}, \Sigma)$, the ASL_d6dafn or ASL_r6dafn obtains the coefficients of the linear discriminant function of $\boldsymbol{u}$ defined by the following equation.

$$ y^{(p)}(\boldsymbol{u}) = {\boldsymbol{\nu_k}}^T \Sigma^{-1} \boldsymbol{u} - \frac{1}{2} {\boldsymbol{\nu_k}}^T \Sigma^{-1} \boldsymbol{\nu_k} \quad (p = 1, 2, \cdots, g) $$

It also obtains the quantity D^2, which is defined by the following equation.

$$ D^2 = \sum_{i=1}^{m} \sum_{j=1}^{m} {\sigma_{i,j}}^{-1} \sum_{l=1}^{g} n_k ({\overline{x}_{\cdot i}}^{(k)} - \overline{x}_{\cdot i})({\overline{x}_{\cdot j}}^{(k)} - \overline{x}_{\cdot j}) $$

Here, $\overline{x}_{\cdot i}$, which represents the mean of the variates over all groups, is defined by the following equation.

$$ \overline{x}_{\cdot i} = \sum_{k=1}^{g} \frac{n_k {\overline{x}_{\cdot i}}^{(k)}}{\sum_{k=1}^{g} n_k} $$

${\sigma_{i,j}}^{-1}$ is an element of Σ^{-1}. The function also obtains the maximum value ${y_{p_m}}^{(l,k)}$ related to $p = 1, 2, \cdots, g$ of the value $y^{(p)}(\boldsymbol{u}^{(l,k)})$ of the discriminant function of ${u_i}^{(l,k)} = {x_{l,i}}^{(k)}$ $(i = 1, 2, \cdots, m)$ and the value ${p_m}^{(l,k)}$ of p at that time. In addition, it obtains the maximum probability of the discriminant function, which is defined by the following equation.

$$ P^{(l,k)} = \frac{1}{\sum_{k=1}^{g} \exp(y^{(k)}(\boldsymbol{u}^{(l,k)}) - {y_{p_m}}^{(l,k)})} \quad (l = 1, 2, \cdots, n_k; k = 1, 2, \cdots, g) $$

(2) **Usage**

Double precision:

```
ierr = ASL_d6dafn  (a, ma, m, n, k, x1, mx1, c, tm, &dist, co, mco, p, num, iw, w1);
```

Single precision:

```
ierr = ASL_r6dafn  (a, ma, m, n, k, x1, mx1, c, tm, &dist, co, mco, p, num, iw, w1);
```

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	a	{D∗ / R∗}	ma×m	Input	Observation data matrix $(x_{l,i}{}^{(k)})$ (See Note (a))
2	ma	I	1	Input	Adjustable dimension of array a
3	m	I	1	Input	Number of variates m
4	n	I*	k	Input	Number of observed values of each group (n_k)
5	k	I	1	Input	Number of groups g
6	x1	{D∗ / R∗}	mx1×k	Input	Mean of variates of each group $(\overline{x}_{\cdot i}^{(k)})$(See Note (a))
				Output	Discriminant function value $y^{(g)}(\boldsymbol{u}^{(i,g)})$ (See Note (a))
7	mx1	I	1	Input	Adjustable dimension of array x1
8	c	{D∗ / R∗}	mx1×m	Input	Variance-covariance matrix Σ^{-1}
9	tm	{D∗ / R∗}	m	Input	Mean of variates over all groups $\overline{x}_{\cdot i}$
10	dist	{D∗ / R∗}	1	Output	D^2 value
11	co	{D∗ / R∗}	mco×k	Output	Discriminant function coefficients $\boldsymbol{\nu_k}^T\Sigma^{-1}$ and $-\frac{1}{2}\boldsymbol{\nu_k}^T\Sigma^{-1}\boldsymbol{\nu_k}$ (See Note (a))
12	mco	I	1	Input	Adjustable dimension of array co
13	p	{D∗ / R∗}	See Contents	Output	Probability $P^{(l,k)}$ $(l = 1,2,\cdots,n_k; k = 1,2,\cdots,g)$ related to the maximum discriminant function of each sample over all groups (See Note (a)) **Size:** n[0] + ··· + n[k − 1]
14	num	I*	See Contents	Output	Number $p_m{}^{(l,k)}$ $(l = 1,2,\cdots,n_k; k = 1,2,\cdots,g)$ of the discriminant function having the largest probability (See Note (a)) **Size:** n[0] + ··· + n[k − 1]
15	iw	I*	m	Work	Work area

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
16	w1	$\left\{\begin{matrix}\text{D}* \\ \text{R}*\end{matrix}\right\}$	m+2	Work	Work area
17	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) $1 \leq \text{m} \leq \text{mx1}$

(b) $\text{k} \geq 2$

(c) $\text{n}[\text{i}-1] \geq 2(\text{i} = 1, \cdots, \text{k})$

(d) $\text{n}[0] + \cdots + \text{n}[\text{k}-1] \leq \text{ma}$

(e) $\text{mco} \geq \text{m} + 1$

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
3000	Restriction (a) was not satisfied.	Processing is aborted.
3010	Restriction (b) was not satisfied.	
3020	Restriction (c) was not satisfied.	
3030	Restriction (d) was not satisfied.	
3040	Restriction (e) was not satisfied.	

(6) **Notes**

(a) Consider g groups for which there are m variates and n_k $(k = 1, 2, \cdots, g)$ observed values of each variate and assume that each observed value is given by ${x_{l,i}}^{(k)}$ $(l = 1, 2, \cdots, n_k; i = 1, 2, \cdots, m; k = 1, 2, \cdots, g)$. The observation data is stored in array a as the following kind of real matrix (two-dimensional array type) (See Appendix A).

$$\begin{bmatrix}
{x_{1,1}}^{(1)} & {x_{1,2}}^{(1)} & \cdots & {x_{1,m}}^{(1)} \\
{x_{2,1}}^{(1)} & {x_{2,2}}^{(1)} & \cdots & {x_{2,m}}^{(1)} \\
\vdots & \vdots & \cdots & \vdots \\
{x_{n_1,1}}^{(1)} & {x_{n_1,2}}^{(1)} & \cdots & {x_{n_1,m}}^{(1)} \\
{x_{1,1}}^{(2)} & {x_{1,2}}^{(2)} & \cdots & {x_{1,m}}^{(2)} \\
\vdots & \vdots & \cdots & \vdots \\
{x_{n_2,1}}^{(2)} & {x_{n_2,2}}^{(2)} & \cdots & {x_{n_2,m}}^{(2)} \\
\vdots & \vdots & \cdots & \vdots \\
{x_{1,1}}^{(g)} & {x_{1,2}}^{(g)} & \cdots & {x_{1,m}}^{(g)} \\
\vdots & \vdots & \cdots & \vdots \\
{x_{n_g,1}}^{(g)} & {x_{n_g,2}}^{(g)} & \cdots & {x_{n_g,m}}^{(g)}
\end{bmatrix}$$

The means of the variates of each group are stored in array x1 as the real matrix (two-dimensional array type) $E = (e_{i,k})$ $(i = 1, 2, \cdots, m; k = 1, 2, \cdots, g)$, which is defined as follows (See Appendix A).

$$e_{i,k} = \overline{x}_{\cdot i}^{(k)}$$

On output, the data corresponding to the first row of matrix E is replaced by the values of the discriminant function $y^{(p)}(\boldsymbol{u}^{(n_g,g)})$ $(p = 1, 2, \cdots, g)$. Also, the discriminant function coefficients are stored in array co as the real matrix (two-dimensional array type) $C = (c_{i,k})$ $(i = 1, 2, \cdots, m+1; k = 1, 2, \cdots, g)$, which is defined as follows (See Appendix A).

$$c_{i,k} = (\boldsymbol{\nu_k}^T \Sigma^{-1})_i \ (i = 1, 2, \cdots, m)$$

$$c_{m+1,k} = (-\frac{1}{2}\boldsymbol{\nu_k}^T \Sigma^{-1} \boldsymbol{\nu_k})$$

The input time data of arrays c and x1 used by this function can be generated from array a by using the function 4.3.2 $\left\{\begin{matrix}\text{ASL\_d2vcgr}\\ \text{ASL\_r2vcgr}\end{matrix}\right\}$. (Where, t = n[0] + $\cdots$ + n[k − 1])

9.5.2 ASL_d6dasc, ASL_r6dasc Discriminant Function Scores

(1) **Function**

Assume there are g groups, the number of observed values consisting of m variates of each group is n_k $(k = 1, 2, \cdots, g)$, the variance-covariance matrix over all groups is Σ, and the populations for the observed values are normal populations of order m represented by $N(\boldsymbol{\nu_1}, \Sigma), \cdots, N(\boldsymbol{\nu_k}, \Sigma)$. Given the observed values ${x_{l,i}}^{(k)}$ $(l = 1, 2, \cdots, n_k; i = 1, 2, \cdots, m; k = 1, 2, \cdots, g)$ of the g groups and the coefficients $C = (c_{i,k})$ $(i = 1, 2, \cdots, m+1; k = 1, 2, \cdots, g)$ of the linear discriminant function of the m -dimensional vector $\boldsymbol{u}$ defined by the following equation:

$$y^{(p)}(\boldsymbol{u}) = {\boldsymbol{\nu_k}}^T \Sigma^{-1} \boldsymbol{u} - \frac{1}{2} {\boldsymbol{\nu_k}}^T \Sigma^{-1} \boldsymbol{\nu_k} \quad (p = 1, 2, \cdots, g; k = 1, 2, \cdots, g)$$

where the $c_{i,k}$ are defined as follows:

$$c_{i,k} = ({\boldsymbol{\nu_k}}^T \Sigma^{-1})_i \ (i = 1, 2, \cdots, m)$$

$$c_{m+1,k} = (-\frac{1}{2} {\boldsymbol{\nu_k}}^T \Sigma^{-1} \boldsymbol{\nu_k})$$

the ASL_d6dasc or ASL_r6dasc obtains the value (discriminant score) ${z_{l,i}}^{(p)} = y^{(p)}(\boldsymbol{u}^{(l,k)})$ $(p = 1, 2, \cdots, g)$ of the discriminant function corresponding to each observed value ${u_i}^{(l,k)} = {x_{l,i}}^{(k)}$ $(i = 1, 2, \cdots, m)$.

(2) **Usage**

Double precision:

```
ierr = ASL_d6dasc  (a, ma, m, n, k, co, mco, z);
```

Single precision:

```
ierr = ASL_r6dasc  (a, ma, m, n, k, co, mco, z);
```

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	a	{ D∗ / R∗ }	ma×m	Input	$({x_{l,i}}^{(k)})$ (See Note (a))
2	ma	I	1	Input	Adjustable dimension of array a
3	m	I	1	Input	Number of variates m
4	n	I*	k	Input	Number of observed values of each group (n_k)
5	k	I	1	Input	Number of groups g
6	co	{ D∗ / R∗ }	mco×k	Input	Discriminant function coefficient $c_{i,k}$ (See Note (a))

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
7	mco	I	1	Input	Adjustable dimension of array co
8	z	$\left\{\begin{matrix}\text{D}*\\ \text{R}*\end{matrix}\right\}$	ma×k	Output	Discriminant function score $y^{(p)}(\boldsymbol{u}^{(l,k)})$
9	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) $2 \leq \text{m} + 1 \leq \text{mco}$

(b) $\text{k} \geq 2$

(c) $\text{n}[\text{i} - 1] \geq 2 \quad (\text{i} = 1, \cdots, \text{k})$

(d) $\text{n}[0] + \cdots + \text{n}[\text{k} - 1] \leq \text{ma}$

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
3000	Restriction (a) was not satisfied.	Processing is aborted.
3010	Restriction (b) was not satisfied.	
3020	Restriction (c) was not satisfied.	
3030	Restriction (d) was not satisfied.	

(6) **Notes**

(a) Consider g groups for which there are m variates and n_k $(k = 1, 2, \cdots, g)$ observed values of each variate and assume that each observed value is given by ${x_{l,i}}^{(k)}$ $(l = 1, 2, \cdots, n_k; i = 1, 2, \cdots, m; k = 1, 2, \cdots, g)$. The observation data is stored in array a as the following kind of real matrix (two-dimensional array type).

$$\begin{bmatrix} {x_{1,1}}^{(1)} & {x_{1,2}}^{(1)} & \cdots & {x_{1,m}}^{(1)} \\ {x_{2,1}}^{(1)} & {x_{2,2}}^{(1)} & \cdots & {x_{2,m}}^{(1)} \\ \vdots & \vdots & \cdots & \vdots \\ {x_{n_1,1}}^{(1)} & {x_{n_1,2}}^{(1)} & \cdots & {x_{n_1,m}}^{(1)} \\ {x_{1,1}}^{(2)} & {x_{1,2}}^{(2)} & \cdots & {x_{1,m}}^{(2)} \\ \vdots & \vdots & \cdots & \vdots \\ {x_{n_2,1}}^{(2)} & {x_{n_2,2}}^{(2)} & \cdots & {x_{n_2,m}}^{(2)} \\ \vdots & \vdots & \cdots & \vdots \\ {x_{1,1}}^{(g)} & {x_{1,2}}^{(g)} & \cdots & {x_{1,m}}^{(g)} \\ \vdots & \vdots & \cdots & \vdots \\ {x_{n_g,1}}^{(g)} & {x_{n_g,2}}^{(g)} & \cdots & {x_{n_g,m}}^{(g)} \end{bmatrix}$$

Also, the discriminant function coefficients are stored in array co as the real matrix (two-dimensional array type) $C = (c_{i,k})$ $(i = 1, 2, \cdots, m + 1; k = 1, 2, \cdots, g)$, which is defined as follows.

$$c_{i,k} = ({\boldsymbol{\nu_k}}^T \Sigma^{-1})_i \ (i = 1, 2, \cdots, m)$$

$$c_{m+1,k} = (-\frac{1}{2} {\boldsymbol{\nu_k}}^T \Sigma^{-1} \boldsymbol{\nu_k})$$

The values (discriminant scores) ${z_{l,i}}^{(p)} = y^{(p)}(\boldsymbol{u}^{(l,k)})$ $(p = 1, 2, \cdots, g)$ of the discriminant functions corresponding to the observed values ${u_i}^{(l,k)} = {x_{l,i}}^{(k)}$ $(i = 1, 2, \cdots, m)$ are stored in array z as a real matrix (two-dimensional array type), which is defined as follows.

$$\begin{bmatrix}
{z_{1,1}}^{(1)} & {z_{1,1}}^{(2)} & \cdots & {z_{1,1}}^{(g)} \\
{z_{2,1}}^{(1)} & {z_{2,1}}^{(2)} & \cdots & {z_{2,1}}^{(g)} \\
\vdots & \vdots & \cdots & \vdots \\
{z_{n_1,1}}^{(1)} & {z_{n_1,1}}^{(2)} & \cdots & {z_{n_1,1}}^{(g)} \\
{z_{1,2}}^{(1)} & {z_{1,2}}^{(2)} & \cdots & {z_{1,2}}^{(g)} \\
\vdots & \vdots & \cdots & \vdots \\
{z_{n_2,2}}^{(1)} & {z_{n_2,2}}^{(2)} & \cdots & {z_{n_2,2}}^{(g)} \\
\vdots & \vdots & \cdots & \vdots \\
{z_{1,g}}^{(1)} & {z_{1,g}}^{(2)} & \cdots & {z_{1,g}}^{(g)} \\
\vdots & \vdots & \cdots & \vdots \\
{z_{n_g,g}}^{(1)} & {z_{n_g,g}}^{(2)} & \cdots & {z_{n_g,g}}^{(g)}
\end{bmatrix}$$

For the method to store the matrix data, see Appendix A.

(7) **Example**

(a) Problem

Read observation data consisting of three groups and obtain the variance-covariance matrix. Based on this variance-covariance matrix, obtain the generalized distance of Mahalanobis (D^2), discriminant function, criteria values for discriminating, and discriminant scores.

(b) Input data

Observation data matrix:

$$A = \begin{bmatrix}
10.0 & 3.0 & 7.0 \\
11.0 & 5.0 & 8.0 \\
12.0 & 7.0 & 6.0 \\
14.0 & 4.0 & 9.0 \\
17.0 & 12.0 & 8.0 \\
18.0 & 11.0 & 6.0 \\
18.0 & 13.0 & 7.0 \\
11.0 & 4.0 & 11.0 \\
12.0 & 6.0 & 12.0 \\
13.0 & 8.0 & 10.0 \\
15.0 & 5.0 & 6.0 \\
18.0 & 10.0 & 13.0
\end{bmatrix}$$

n[0] = 4,
n[1] = 3,
n[2] = 5,
k = 3, m = 3, ma = 12,
mx1 = 3 and mco = 4.

(c) Main program

```
/*      C interface example for ASL_d6dasc, STAT_d6dafn */

#include <stdio.h>
#include <stdlib.h>
#include <asl.h>

int main()
{
    double *a;
    int ma, m;
    int *n;
    int k;
    double *co;
    int mco;
    double *z;

    double *x1;
    int mx1;
    double *c, *tm;
    double dist;
    double *p;
    int *num, *iw, *ns;
    double *w1,*wk;
    int ierr;

    int i,j,l,nt,l1,l2,isw;
    FILE *fp;


    fp = fopen( "d6dasc.dat", "r" );
    if( fp == NULL )
    {
        printf( "file open error\n" );
        return -1;
    }

    printf( "    *** ASL_d6dasc ***\n" );
    printf( "\n     ** Input **\n\n" );

    fscanf( fp, "%d", &ma );
    fscanf( fp, "%d", &m );
    fscanf( fp, "%d", &k );

    mx1=m;
    mco=mx1+1;
    isw=0;

    printf( "\tma =%3d\n", ma );
    printf( "\tm  =%3d\n", m );
    printf( "\tk  =%3d\n", k );
    printf( "\tmx1=%3d\n", mx1 );
    printf( "\tmco=%3d\n", mco );
    printf( "\tisw=%3d\n\n", isw );

    a = ( double * )malloc((size_t)( sizeof(double) * (ma*m) ));
    if( a == NULL )
    {
        printf( "no enough memory for array a\n" );
        return -1;
    }

    x1 = ( double * )malloc((size_t)( sizeof(double) * (mx1*k) ));
    if( x1 == NULL )
    {
        printf( "no enough memory for array x1\n" );
        return -1;
    }

    tm = ( double * )malloc((size_t)( sizeof(double) * m ));
    if( tm == NULL )
    {
        printf( "no enough memory for array tm\n" );
        return -1;
    }

    num = ( int * )malloc((size_t)( sizeof(int) * ma ));
    if( num == NULL )
    {
        printf( "no enough memory for array num\n" );
        return -1;
    }

    n = ( int * )malloc((size_t)( sizeof(int) * k ));
    if( n == NULL )
    {
        printf( "no enough memory for array n\n" );
        return -1;
    }

    iw = ( int * )malloc((size_t)( sizeof(int) * m ));
```

```
if( iw == NULL )
{
    printf( "no enough memory for array iw\n" );
    return -1;
}

c = ( double * )malloc((size_t)( sizeof(double) * (mx1*m) ));
if( c == NULL )
{
    printf( "no enough memory for array c\n" );
    return -1;
}

co = ( double * )malloc((size_t)( sizeof(double) * (mco*k) ));
if( co == NULL )
{
    printf( "no enough memory for array co\n" );
    return -1;
}

p = ( double * )malloc((size_t)( sizeof(double) * ma ));
if( p == NULL )
{
    printf( "no enough memory for array p\n" );
    return -1;
}

z = ( double * )malloc((size_t)( sizeof(double) * (ma*k) ));
if( z == NULL )
{
    printf( "no enough memory for array z\n" );
    return -1;
}

w1 = ( double * )malloc((size_t)( sizeof(double) * (m+2) ));
if( w1 == NULL )
{
    printf( "no enough memory for array w1\n" );
    return -1;
}

ns = ( int * )malloc((size_t)( sizeof(int) * k ));
if( ns == NULL )
{
    printf( "no enough memory for array ns\n" );
    return -1;
}

wk = ( double * )malloc((size_t)( sizeof(double) * (m*m*k+m) ));
if( wk == NULL )
{
    printf( "no enough memory for array wk\n" );
    return -1;
}

for( i=0 ; i<ma ; i++ )
{
    for( j=0 ; j<m ; j++ )
    {
        fscanf( fp, "%lf", &a[i+ma*j] );
    }
}

for( i=0 ; i<k ; i++ )
{
    fscanf( fp, "%d", &n[i] );
}

fclose( fp );

printf( "\tNumber of observations(each group)\n" );
printf( "\t" );
for( i=0 ; i<k ; i++ )
{
    printf( "%6d", n[i] );
}
printf( "\n\n" );

printf( "\tObservation Data\n" );
nt=0;
for( i=0 ; i<k ; i++ )
{
    nt+=n[i];
    printf( "\tGroup %2d\n", i );
    for( l=0 ; l<n[i] ; l++ )
    {
        printf( "\t%2d  ", nt-n[i]+l );
        for( j=0 ; j<m ; j++ )
        {
            printf( "%8.3g", a[nt-n[i]+l+ma*j] );
        }
        printf( "\n" );
```

```
    }
    printf( "\n" );
}

ierr = ASL_d2vcgr(a, ma, m, n, k, ns, tm, x1, mx1, c, mx1, isw, wk);

printf( "     ** Output (ASL_d2vcgr) **\n\n" );
printf( "\tierr = %6d\n\n", ierr );

printf( "\tMean of variables\n" );
for( j=0 ; j<k ; j++ )
{
    printf( "\tGroup %2d\n\t", j );
    for( i=0 ; i<m ; i++ )
    {
        printf( "%8.3g", x1[i+mx1*j] );
    }
    printf( "\n\n" );
}

printf( "\tTotal mean of variables\n\t" );
for( i=0 ; i<m ; i++ )
{
    printf( "%8.3g", tm[i] );
}
printf( "\n\n" );

printf( "\tVariance covariance matrix\n" );
for( i=0 ; i<m ; i++ )
{
    printf( "\t" );
    for( j=0 ; j<m ; j++ )
    {
        printf( "%8.3g", c[i+mx1*j] );
    }
    printf( "\n" );
}
printf( "\n" );

ierr = ASL_d6dafn(a, ma, m, n, k, x1, mx1, c, tm,
         &dist, co, mco, p, num, iw, w1);

printf( "     ** Output (ASL_d6dafn) **\n\n" );
printf( "\tierr = %6d\n\n", ierr );

printf( "\tMaharanobis distance\n" );
printf( "\t\tdist = %8.3g\n\n", dist );

printf( "\tDiscriminant coefficient\n" );
for( j=0 ; j<m ; j++ )
{
    printf( "\t          " );
    for( i=0 ; i<k ; i++ )
    {
        printf( "%8.3g", co[j+mco*i] );
    }
    printf( "\n" );
}
printf( "\t Constant " );
for( i=0 ; i<k ; i++ )
{
    printf( "%8.3g", co[m+mco*i] );
}
printf( "\n\n" );

printf( "\tEvaluation of classification\n" );
l1=0;
l2=n[0];
for( i=0 ; i<k ; i++ )
{
    printf( "\n\tGroup %2d\n" , i );
    printf( "\tMaximum probability  Maximum function no\n" );
    for( j=l1 ; j<l2 ; j++ )
    {
        printf( "\t%8.3g          %6d\n", p[j], num[j] );
    }
    l1+=n[i];
    l2+=n[i+1];
}
printf( "\n" );

ierr = ASL_d6dasc(a, ma, m, n, k, co, mco, z);

printf( "     ** Output (ASL_d6dasc) **\n\n" );
printf( "\tierr = %6d\n\n", ierr );

printf( "\tDiscriminant score\n" );
printf( "\t" );
```

```
    for( j=0 ; j<k ; j++ )
    {
        printf( " Group %1d  ", j );
    }
    printf( "\n" );
    for( i=0 ; i<nt ; i++ )
    {
        printf( "\t" );
        for( j=0 ; j<k ; j++ )
        {
            printf( "%8.3g  ", z[i+ma*j] );
        }
        printf( "\n" );
    }

    free( num );
    free( n );
    free( iw );
    free( a );
    free( x1 );
    free( tm );
    free( c );
    free( co );
    free( p );
    free( z );
    free( w1 );
    free( ns );
    free( wk );

    return 0;
}
```

(d) Output results

```
*** ASL_d6dasc ***

 ** Input **

ma = 12
m  =  3
k  =  3
mx1=  3
mco=  4
isw=  0

Number of observations(each group)
     4     3     5

Observation Data
Group  0
 0        10       3       7
 1        11       5       8
 2        12       7       6
 3        14       4       9

Group  1
 4        17      12       8
 5        18      11       6
 6        18      13       7

Group  2
 7        11       4      11
 8        12       6      12
 9        13       8      10
10        15       5       6
11        18      10      13

 ** Output (ASL_d2vcgr) **

ierr =      0

Mean of variables
Group  0
    11.8    4.75     7.5

Group  1
    17.7      12       7

Group  2
    13.8     6.6    10.4

Total mean of variables
    14.1    7.33    8.58

Variance covariance matrix
    4.47    2.37   0.433
    2.37    3.77    1.14
   0.433    1.14    4.02

 ** Output (ASL_d6dafn) **

ierr =      0
```

```
Maharanobis distance
    dist =     39.3

Discriminant coefficient
              3.13    3.51    3.49
             -1.28   0.611   -1.22
              1.89    1.19    2.56
 Constant    -22.4   -38.9   -33.3

Evaluation of classification

Group  0
Maximum probability  Maximum function no
   0.922               1
   0.788               1
   0.849               1
   0.592               3

Group  1
Maximum probability  Maximum function no
   0.997               2
   0.998               2
       1               2

Group  2
Maximum probability  Maximum function no
    0.65               3
   0.854               3
   0.708               3
   0.765               1
   0.982               3

 ** Output (ASL_d6dasc) **

ierr =      0

Discriminant score
 Group 0   Group 1   Group 2
    18.2      6.43      15.8
    20.7      12.3      19.4
    17.5      14.7      15.3
    33.3      23.5      33.6
    30.5      37.7      31.8
    31.1      38.2      31.4
    30.4      40.6      31.5
    27.7      15.3      28.3
    30.1      21.2      31.9
    26.9      23.6      27.8
    29.4        24      28.2
    45.6      45.9      50.5
```

9.6 CLUSTER ANALYSIS

9.6.1 ASL_d6clds, ASL_r6clds Dissimilarity Measures

(1) **Function**

If individuals or variates having n characteristics are to be used as classification subjects when the (individual) × (variate) multivariate characteristic value data matrix (a_{ik}) or (a_{ki}) $(i = 1, 2, \cdots, n; k = 1, 2, \cdots, p)$ is given, the ASL_d6clds or ASL_r6clds obtains the dissimilarity measure d_{ij} $(i, j = 1, 2, \cdots, n)$ between the i-th and j-th individual or variate by using the measures described below.

- Euclidean quadratic distance

$$d_{ij} = \sum_{k=1}^{p} (a_{ik} - a_{jk})^2 \ (i, j = 1, \cdots, n)$$

- Standardized Euclidean quadratic distance

$$d_{ij} = \sum_{k=1}^{p} \frac{(a_{ik} - a_{jk})^2}{s_k^2} \ (i, j = 1, \cdots, n)$$

Here, s_k^2, which is the variance of the variate, is defined by the following equation.

$$s_k^2 = \frac{1}{n-1} \sum_{l=1}^{n} (a_{lk} - \bar{a_k})^2 (\bar{a_k} = \frac{1}{n} \sum_{l=1}^{n} a_{lk})$$

This is the same as obtaining the Euclidean quadratic distance when the variance of each variate is standardized to 1.

- Generalized distance of Mahalanobis

$$d_{ij} = \sum_{k=1}^{p} \sum_{m=1}^{p} (a_{ik} - a_{jk}) v_{km} (a_{im} - a_{jm}) \ (i, j = 1, \cdots, n)$$

Here, v_{km} is the (k, m) element of the inverse matrix of the variance-covariance matrix of the individuals and variates.

- Minkowski distance

$$d_{ij} = \{ \sum_{k=1}^{p} \mid a_{ik} - a_{jk} \mid^r \}^{1/r} \ (r \geq 1.0; i, j = 1, \cdots, n)$$

(2) **Usage**

Double precision:

ierr = ASL_d6clds (a, lx, ly, nx, ny, ml, diss, isw, w1);

Single precision:

ierr = ASL_r6clds (a, lx, ly, nx, ny, ml, diss, isw, w1);

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	a	{D∗ / R∗}	lx×ly	Input	Observation data a_{ik} or a_{ki} $(i = 1, 2, \cdots, n;$ $k = 1, 2, \cdots, p)$ (See Note (a))
2	lx	I	1	Input	Adjustable dimension of array a
3	ly	I	1	Input	Second dimension of array a
4	nx	I	1	Input	Number of rows of observation data matrix (n or p)
5	ny	I	1	Input	Number of columns of observation data matrix (p or n)
6	ml	I	1	Input	max(lx, ly)
7	diss	{D∗ / R∗}	See Contents	Input	When isw=14 or 24 (Minkowski distance), enter the value of r $(r \geq 1.0)$ for diss[0]. Otherwise, this setting is unnecessary. **Size:** When nx=n, the size is (ml×nx) When ny=n, the size is (ml×ny)
				Output	Dissimilarity measure matrix (real symmetric matrix)
8	isw	I	1	Input	Dissimilarity measure calculation processing switch isw=11 or 21: Euclidean quadratic distance isw=12 or 22: Standardized Euclidean quadratic distance isw=13 or 23: (Generalized) distance of Mahalanobis isw=14 or 24: Minkowski distance For isw=1x, nx=n For isw=2x, ny=n

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
9	w1	$\left\{\begin{matrix}\text{D}*\\ \text{R}*\end{matrix}\right\}$	See Contents	Work	Work area **Size:** When nx=n, the size is ny $\times$ (ml + 1) + 2 When ny=n, the size is nx $\times$ (ml + 1) + 2
10	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) $1 < \text{nx} \leq \text{lx}, 1 < \text{ny} \leq \text{ly}$

(b) $\text{ml} = \max(\text{lx}, \text{ly})$

(c) $\text{isw} \in \{11, 12, 13, 14, 21, 22, 23, 24\}$

(d) When isw = 14 or 24, $\text{diss}[1] \geq 1.0$

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
3000	Restriction (a) was not satisfied.	Processing is aborted.
3010	Restriction (b) was not satisfied.	
3020	Restriction (c) was not satisfied.	
3030	Restriction (d) was not satisfied.	
4000	For the entered observation data, the calculation cannot be performed using the (generalized) distance of Mahalanobis. (see Section 9.1.1)	

(6) **Notes**

(a) When the Euclidean quadratic distance or Minkowski distance is used for the dissimilarity measure calculation and the units of the observation data subjects differ, the data should be standardized in advance so that a suitable positional relationship between the subjects is reflected (see the example).

(7) **Example**

(a) Problem
Standardize the following observation data matrix:

$$A = \begin{bmatrix} 0.321 & 119 & 6 & 40 & 6 & 6 & 12 & 29 \\ 0.301 & 112 & 9 & 38 & 4 & 2 & 3 & 31 \\ 0.288 & 133 & 15 & 53 & 4 & 5 & 12 & 30 \\ 0.280 & 112 & 9 & 47 & 4 & 2 & 10 & 24 \\ 0.261 & 109 & 3 & 21 & 1 & 3 & 13 & 21 \\ 0.256 & 107 & 8 & 34 & 4 & 2 & 17 & 38 \\ 0.253 & 95 & 16 & 57 & 4 & 6 & 7 & 37 \\ 0.250 & 100 & 13 & 46 & 4 & 3 & 13 & 37 \\ 0.249 & 92 & 17 & 48 & 6 & 2 & 7 & 23 \\ 0.227 & 88 & 12 & 45 & 4 & 0 & 3 & 33 \end{bmatrix}$$

let the number of rows be the number of subjects, and obtain the dissimilarity measure matrix by using the Euclidean quadratic distance.

(b) Input data
Observation data matrix A, lx = 11, ly = 9, nx = 10, ny = 8, ml = 11 and isw = 11.

(c) Main program

```
/*      C interface example for ASL_d6clds */

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <asl.h>

int main()
{
    double *a;
    int lx, ly;
    int nx, ny;
    int ml;
    double *diss;
    int isw;
    double *w1;
    int ierr;
    int i,j;
    double sum,*av,*c;

    FILE *fp;

    lx=11;
    ly= 9;
    nx=10;
    ny= 8;
    ml=11;
    isw=11;


    fp = fopen( "d6clds.dat", "r" );
    if( fp == NULL )
    {
        printf( "file open error\n" );
        return -1;
    }

    printf( "    *** ASL_d6clds ***\n" );
    printf( "\n     ** Input **\n\n" );

    a = ( double * )malloc((size_t)( sizeof(double) * (lx*ly) ));
    if( a == NULL )
    {
        printf( "no enough memory for array a\n" );
        return -1;
    }

    diss = ( double * )malloc((size_t)( sizeof(double) * (ml*nx) ));
    if( diss == NULL )
    {
        printf( "no enough memory for array diss\n" );
        return -1;
    }

    w1 = ( double * )malloc((size_t)( sizeof(double) * (ny*(ml+1)+2) ));
    if( w1 == NULL )
    {
        printf( "no enough memory for array w1\n" );
        return -1;
    }

    av = ( double * )malloc((size_t)( sizeof(double) * ny ));
    if( av == NULL )
    {
        printf( "no enough memory for array av\n" );
        return -1;
    }

    c = ( double * )malloc((size_t)( sizeof(double) * ny ));
    if( c == NULL )
    {
        printf( "no enough memory for array c\n" );
        return -1;
    }

    printf( "\tisw=%3d\n", isw );
    printf( "\tlx =%3d,  ly =%3d\n", lx, ly );
    printf( "\tnx =%3d,  ny =%3d\n", nx, ny );
    printf( "\tml =%3d\n\n", ml );

    printf( "\t      A      B      C      D");
```

```
        printf( "      E      F      G      H\n");
        printf( "\t----------------------------");
        printf( "----------------------------\n");

        for( i=0 ; i<nx ; i++ )
        {
            printf( "\t" );
            for( j=0 ; j<ny ; j++ )
            {
                fscanf( fp, "%lf", &a[i+lx*j] );
                printf( "%7.3g", a[i+lx*j] );
            }
            printf( "\n" );
        }

        fclose( fp );

        for( j=0 ; j<ny ; j++ )
        {
            sum=0.0;
            for( i=0 ; i<nx ; i++ )
            {
                sum += a[i+lx*j];
            }
            av[j] = sum / (double) nx;
        }
        for( i=0 ; i<ny ; i++ )
        {
            sum=0.0;
            for( j=0 ; j<nx ; j++ )
            {
                sum += (a[j+lx*i]-av[i])*(a[j+lx*i]-av[i]);
            }
            c[i] = sum / ( (double) nx-1.0);
        }
        for( i=0 ; i<nx ; i++ )
        {
            for( j=0 ; j<ny ; j++ )
            {
                a[i+lx*j] = (a[i+lx*j]-av[j])/sqrt(c[j]);
            }
        }

        ierr = ASL_d6clds(a, lx, ly, nx, ny, ml, diss, isw, w1);

        printf( "\n     ** Output **\n\n" );
        printf( "\tierr = %6d\n\n", ierr );

        printf( "\tMatrix diss\n\n" );
        for( i=0 ; i<nx ; i++ )
        {
            printf( "\t" );
            for( j=0 ; j<nx ; j++ )
            {
                printf( "%6.1f", diss[i+ml*j] );
            }
            printf( "\n" );
        }

        free( a );
        free( diss );
        free( w1 );
        free( av );
        free( c );

        return 0;
    }
```

(d) Output results

```
*** ASL_d6clds ***

 ** Input **

isw= 11
lx = 11,  ly =  9
nx = 10,  ny =  8
ml = 11

      A      B      C      D      E      F      G      H
-------------------------------------------------------------
  0.321    119      6     40      6      6     12     29
  0.301    112      9     38      4      2      3     31
  0.288    133     15     53      4      5     12     30
   0.28    112      9     47      4      2     10     24
  0.261    109      3     21      1      3     13     21
  0.256    107      8     34      4      2     17     38
  0.253     95     16     57      4      6      7     37
   0.25    100     13     46      4      3     13     37
  0.249     92     17     48      6      2      7     23
```

```
  0.227      88      12      45       4       0       3      33

 ** Output **

ierr =       0

Matrix diss

   0.0  11.4  10.3  10.4  26.3  16.2  21.5  17.1  23.1  33.9
  11.4   0.0  12.7   4.9  19.0  13.4  16.1  11.4  14.2  12.0
  10.3  12.7   0.0   8.0  28.7  15.9  12.3  10.8  18.4  27.2
  10.4   4.9   8.0   0.0  14.4  10.0  14.8   7.9   9.0  12.6
  26.3  19.0  28.7  14.4   0.0  16.4  37.2  22.9  33.5  28.9
  16.2  13.4  15.9  10.0  16.4   0.0  17.8   3.9  19.9  15.9
  21.5  16.1  12.3  14.8  37.2  17.8   0.0   5.8  12.4  13.7
  17.1  11.4  10.8   7.9  22.9   3.9   5.8   0.0  10.5   9.0
  23.1  14.2  18.4   9.0  33.5  19.9  12.4  10.5   0.0   8.6
  33.9  12.0  27.2  12.6  28.9  15.9  13.7   9.0   8.6   0.0
```

9.6.2 ASL_d6clan, ASL_r6clan
Cluster Analysis (Input Dissimilarity Measure or Similarity Measure)

(1) **Function**

Given the dissimilarity measure d_{ij} or similarity measure $1.0 - d_{ij}$ $(i, j = 1, 2, \cdots, n)$, the ASL_d6clan or ASL_r6clan performs a cluster analysis based on this measure. When a new cluster t is created by merging cluster p and cluster q, the following measures are used as the dissimilarity measure d_{tr} between cluster t and a separate arbitrary cluster r. Here, n_p represents the number of subjects contained in cluster p.

- Nearest neighbor method

$$d_{tr} = \min(d_{pr}, d_{qr})$$

- Furtherest neighbor method

$$d_{tr} = \max(d_{pr}, d_{qr})$$

- Group mean method

$$d_{tr} = (n_p d_{pr} + n_q d_{qr})/(n_p + n_q)$$

- Center of gravity method

$$d_{tr} = \frac{n_p}{n_p + n_q} d_{pr} + \frac{n_q}{n_p + n_q} d_{qr} - \frac{n_p n_q}{(n_p + n_q)^2} d_{pq}$$

- Median method

$$d_{tr} = \frac{1}{2} d_{pr} + \frac{1}{2} d_{qr} - \frac{1}{4} d_{pq}$$

- Ward's method

$$d_{tr} = \frac{n_p + n_r}{n_t + n_r} d_{pr} + \frac{n_q + n_r}{n_t + n_r} d_{qr} - \frac{n_r}{n_t + n_r} d_{pq}$$

- Variable method

$$d_{tr} = \frac{1 - \beta}{2} d_{pr} + \frac{1 - \beta}{2} d_{qr} + \beta d_{pq} \ (-\frac{1}{4} \leq \beta \leq 0)$$

The center of gravity method, median method, and Ward's method assume that the dissimilarity measure is given by the (standardized) Euclidean quadratic distance.

(2) **Usage**

Double precision:

ierr = ASL_d6clan (a, lna, nc, ipos, lnp, dis, isw1, isw2, iw);

Single precision:

ierr = ASL_r6clan (a, lna, nc, ipos, lnp, dis, isw1, isw2, iw);

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	a	{D∗ / R∗}	lna×nc	Input	Dissimilarity measure or similarity measure matrix (real symmetric matrix) (two-dimensional array type, upper triangular type) (See Note (a))
				Output	Entered contents are not saved.
2	lna	I	1	Input	Adjustable dimension of array a
3	nc	I	1	Input	Second dimension of array a
4	ipos	I*	lnp× 2	Output	Information indicating cluster merging (See Note (b)) This information indicates that the i-th merge was a merging of the ipos[i − 1]-th cluster and the ipos[i − 1 + lnp]-th cluster (i = 1, ··· , nc − 1)
5	lnp	I	1	Input	Adjustable dimension of array ipos
6	dis	{D∗ / R∗}	nc − 1	Input	When isw2=7, enter the value of β ($-1/4 \leq \beta \leq 0$) for dis[0] (Default value:$\beta = -1/4$). Otherwise, this setting is unnecessary.
				Output	Cluster merging distance information (See Note (c)) This information indicates that clusters were merged using distance dis[i − 1] during the i-th merge (i = 1, ··· , nc − 1)
7	isw1	I	1	Input	Processing switch (See Note (c)) isw1=1: When the dissimilarity measure matrix is given for array a isw1=2: When the similarity measure matrix is given for array a
8	isw2	I	1	Input	Analysis method selection switch isw2=1: Nearest neighbor method isw2=2: Furtherest neighbor method isw2=3: Group mean method isw2=4: Center of gravity method isw2=5: Median method isw2=6: Ward's method isw2=7: Variable method (input β)

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
9	iw	I*	nc	Work	Work area
10	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) $1 < \mathrm{nc} \leq \mathrm{lna}$

(b) lnp≥nc−1

(c) isw1=1 or isw1=2

(d) $1 \leq \mathrm{isw2} \leq 7$

(e) When isw2 = 7, $-1/4 \leq \mathrm{dis}[0] \leq 0$
(Except when entering 1.0 to set the default value)

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
1000	Restriction (e) was not satisfied.	Processing is performed with the default value set.
3000	Restriction (a) was not satisfied.	Processing is aborted.
3010	Restriction (b) was not satisfied.	
3020	Restriction (c) was not satisfied.	
3030	Restriction (d) was not satisfied.	

(6) **Notes**

(a) The matrix assigned for array a must be a real symmetric matrix. This function uses only the upper triangular portion to perform the processing.

(b) Information indicating which cluster was merged with which cluster during which merge is stored in array ipos. However, the lowest cluster number of the two clusters before merging is assigned to the merged cluster. For example, the cluster formed by merging cluster 3 and cluster 5 is named cluster 3.

(c) When the matrix (a_{ij}) given for array a is a similarity measure matrix, the function performs the analysis after internally converting this to a dissimilarity measure matrix by using the following equation.

$$a_{ij} = 1.0 - a_{ij}$$

Therefore, in this case, the distances calculated by using the values that were converted by using the equation shown above are stored in $\mathrm{dis}[\mathrm{i}-1](\mathrm{i} = 1, \cdots, \mathrm{nc}-1)$.

(7) **Example**

(a) Problem

Perform a cluster analysis using the following dissimilarity measure matrix as an index.

$$A = \begin{bmatrix} 0.000 & 11.432 & 10.328 & 10.369 & 26.271 & 16.179 & 21.455 & 17.142 & 23.111 & 33.883 \\ 11.432 & 0.000 & 12.656 & 4.942 & 18.957 & 13.443 & 16.062 & 11.350 & 14.159 & 11.996 \\ 10.328 & 12.656 & 0.000 & 8.019 & 28.694 & 15.942 & 12.338 & 10.771 & 18.421 & 27.228 \\ 10.369 & 4.942 & 8.019 & 0.000 & 14.353 & 10.042 & 14.833 & 7.866 & 9.040 & 12.615 \\ 26.271 & 18.957 & 28.694 & 14.353 & 0.000 & 16.356 & 37.180 & 22.924 & 33.453 & 28.872 \\ 16.179 & 13.443 & 15.942 & 10.042 & 16.356 & 0.000 & 17.771 & 3.919 & 19.886 & 15.906 \\ 21.455 & 16.062 & 12.338 & 14.833 & 37.180 & 17.771 & 0.000 & 5.752 & 12.359 & 13.710 \\ 17.142 & 11.350 & 10.771 & 7.866 & 22.924 & 3.919 & 5.752 & 0.000 & 10.465 & 8.982 \\ 23.111 & 14.159 & 18.421 & 9.040 & 33.453 & 19.886 & 12.359 & 10.465 & 0.000 & 8.556 \\ 33.883 & 11.996 & 27.228 & 12.615 & 28.872 & 15.906 & 13.710 & 8.982 & 8.556 & 0.000 \end{bmatrix}$$

(b) Input data

Dissimilarity measure matrix A, isw1 = 1, isw2 = 3, lna = 11, nc = 10 and lnp = 9.

(c) Main program

```
/*      C interface example for ASL_d6clan */

#include <stdio.h>
#include <stdlib.h>
#include <asl.h>

int main()
{
    double *a;
    int lna;
    int nc;
    int *ipos;
    int lnp;
    double *dis;
    int isw1, isw2;
    int *iw;
    int ierr;
    int i,j;
    FILE *fp;

    lna=11;
    nc =10;
    lnp=9;
    isw1=1;
    isw2=3;

    fp = fopen( "d6clan.dat", "r" );
    if( fp == NULL )
    {
        printf( "file open error\n" );
        return -1;
    }

    printf( "    *** ASL_d6clan ***\n" );
    printf( "\n     ** Input **\n\n" );

    printf( "\tlna  = %4d\n",  lna );
    printf( "\tnc   = %4d\n",   nc );
    printf( "\tlnp  = %4d\n",  lnp );
    printf( "\tisw1 = %4d\n", isw1 );
    printf( "\tisw2 = %4d\n", isw2 );

    a = ( double * )malloc((size_t)( sizeof(double) * (lna*nc) ));
    if( a == NULL )
    {
        printf( "no enough memory for array a\n" );
        return -1;
    }

    dis = ( double * )malloc((size_t)( sizeof(double) * (nc-1) ));
    if( dis == NULL )
    {
        printf( "no enough memory for array dis\n" );
        return -1;
    }

    ipos = ( int * )malloc((size_t)( sizeof(int) * (nc * 2) ));
    if( ipos == NULL )
    {
        printf( "no enough memory for array ipos\n" );
        return -1;
    }
```

```
    iw = ( int * )malloc((size_t)( sizeof(int) * nc ));
    if( iw == NULL )
    {
        printf( "no enough memory for array iw\n" );
        return -1;
    }

    printf( "\n\tArray a\n\n" );
    for( i=0 ; i<nc ; i++ )
    {
        printf( "\t" );
        for( j=0 ; j<nc ; j++ )
        {
            fscanf( fp, "%lf", &a[i+lna*j] );
            printf( "%6.3g", a[i+lna*j] );
        }
        printf( "\n" );
    }

    fclose( fp );

    ierr = ASL_d6clan(a, lna, nc, ipos, lnp, dis, isw1, isw2, iw);

    printf( "\n     ** Output **\n\n" );
    printf( "\tierr = %6d\n\n", ierr );

    for( i=0 ; i<nc-1 ; i++ )
    {
        printf( "\t%6d --- %6d  %8.3g\n",
            ipos[i], ipos[i+lnp], dis[i] );
    }

    free( a );
    free( ipos );
    free( dis );
    free( iw );

    return 0;
}
```

(d) Output results

```
*** ASL_d6clan ***

 ** Input **

lna  =   11
nc   =   10
lnp  =    9
isw1 =    1
isw2 =    3

Array a

     0  11.4  10.3  10.4  26.3  16.2  21.5  17.1  23.1  33.9
  11.4     0  12.7  4.94    19  13.4  16.1  11.3  14.2    12
  10.3  12.7     0  8.02  28.7  15.9  12.3  10.8  18.4  27.2
  10.4  4.94  8.02     0  14.4    10  14.8  7.87  9.04  12.6
  26.3    19  28.7  14.4     0  16.4  37.2  22.9  33.5  28.9
  16.2  13.4  15.9    10  16.4     0  17.8  3.92  19.9  15.9
  21.5  16.1  12.3  14.8  37.2  17.8     0  5.75  12.4  13.7
  17.1  11.3  10.8  7.87  22.9  3.92  5.75     0  10.5  8.98
  23.1  14.2  18.4  9.04  33.5  19.9  12.4  10.5     0  8.56
  33.9    12  27.2  12.6  28.9  15.9  13.7  8.98  8.56     0

 ** Output **

ierr =      0

     6 ---      8      3.92
     2 ---      4      4.94
     9 ---     10      8.56
     1 ---      3      10.3
     1 ---      2      10.6
     6 ---      7      11.8
     6 ---      9      13.6
     1 ---      6      15.9
     1 ---      5      25.2
```

9.6.3 ASL_d6clda, ASL_r6clda
Cluster Analysis (Input Observation Data, Use Dissimilarity Measure)

(1) **Function**

If individuals or variates having n characteristics are to be used as classification subjects when the (individual) × (variate) multivariate characteristic value data matrix (a_{ik}) or (a_{ki}) $(i = 1, 2, \cdots, n;\ k = 1, 2, \cdots, p)$ is given, the ASL_d6clda or ASL_r6clda obtains the dissimilarity measure d_{ij} $(i, j = 1, 2, \cdots, n)$ between the i-th and j-th individual or variate by using the measures described below and performs a cluster analysis using this as an index.

- Euclidean quadratic distance

$$d_{ij} = \sum_{k=1}^{p} (a_{ik} - a_{jk})^2 \ (i, j = 1, \cdots, n)$$

- Standardized Euclidean quadratic distance

$$d_{ij} = \sum_{k=1}^{p} \frac{(a_{ik} - a_{jk})^2}{s_k^2} \ (i, j = 1, \cdots, n)$$

  Here, s_k^2, which is the variance of the variate, is defined by the following equation.

$$s_k^2 = \frac{1}{n-1} \sum_{l=1}^{n} (a_{lk} - \bar{a_k})^2 (\bar{a_k} = \frac{1}{n} \sum_{l=1}^{n} a_{lk})$$

  This is the same as obtaining the Euclidean quadratic distance when the variance of each variate is standardized to 1.

- Generalized distance of Mahalanobis

$$d_{ij} = \sum_{k=1}^{p} \sum_{m=1}^{p} (a_{ik} - a_{jk}) v_{km} (a_{im} - a_{jm}) \ (i, j = 1, \cdots, n)$$

  Here, v_{km} is the v_{km} element of the inverse matrix of the variance-covariance matrix of the individuals and variates.

- Minkowski distance

$$d_{ij} = \{ \sum_{k=1}^{p} \mid a_{ik} - a_{jk} \mid^r \}^{1/r} \ (r \geq 1.0; i, j = 1, \cdots, n)$$

When a new cluster t is created by merging cluster p and cluster q, the following measures are used as the dissimilarity measure d_{tr} between cluster t and a separate arbitrary cluster r. Here, n_p represents the number of subjects contained in cluster p.

- Nearest neighbor method

$$d_{tr} = \min(d_{pr}, d_{qr})$$

- Furtherest neighbor method

$$d_{tr} = \max(d_{pr}, d_{qr})$$

- Group mean method

$$d_{tr} = (n_p d_{pr} + n_q d_{qr}) / (n_p + n_q)$$

- Center of gravity method

$$d_{tr} = \frac{n_p}{n_p + n_q} d_{pr} + \frac{n_q}{n_p + n_q} d_{qr} - \frac{n_p n_q}{(n_p + n_q)^2} d_{pq}$$

- Median method

$$d_{tr} = \frac{1}{2}d_{pr} + \frac{1}{2}d_{qr} - \frac{1}{4}d_{pq}$$

- Ward's method

$$d_{tr} = \frac{n_p + n_r}{n_t + n_r}d_{pr} + \frac{n_q + n_r}{n_t + n_r}d_{qr} - \frac{n_r}{n_t + n_r}d_{pq}$$

- Variable method

$$d_{tr} = \frac{1-\beta}{2}d_{pr} + \frac{1-\beta}{2}d_{qr} + \beta d_{pq} \ (-\frac{1}{4} \leq \beta \leq 0)$$

The center of gravity method, median method, and Ward's method assume that the dissimilarity measure is given by the (standardized) Euclidean quadratic distance.

(2) **Usage**

Double precision:

ierr = ASL_d6clda (a, lx, ly, nx, ny, ml, nc, ipos, lnp, dis, isw1, isw2, iw, w1);

Single precision:

ierr = ASL_r6clda (a, lx, ly, nx, ny, ml, nc, ipos, lnp, dis, isw1, isw2, iw, w1);

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	a	{D∗ / R∗}	lx×ly	Input	Observation data a_{ik} or a_{ki} ($i = 1, 2, \cdots, n$; $k = 1, 2, \cdots, p$) (See Note (a))
2	lx	I	1	Input	Adjustable dimension of array a
3	ly	I	1	Input	Second dimension of array a
4	nx	I	1	Input	Number of rows of observation data matrix (n or p)
5	ny	I	1	Input	Number of columns of observation data matrix (p or n)
6	ml	I	1	Input	max(lx, ly)
7	nc	I	1	Input	Number of objects n (nx or ny)
8	ipos	I*	lnp× 2	Output	Information indicating cluster merging (See Note (b)) This information indicates that the i-th merge was a merging of the ipos[i − 1]-th cluster and the ipos[i − 1]-th cluster (i = $1, \cdots,$ nc − 1)
9	lnp	I	1	Input	Adjustable dimension of array ipos
10	dis	{D∗ / R∗}	nc − 1	Input	When isw1=14 or 24 (Minkowski distance), enter the value of r ($r \geq 1.0$) for dis[0]. When isw2 = 7, enter the value of β ($-1/4 \leq \beta \leq 0$) for dis[1] (Default value: $\beta = -1/4$). Otherwise, this setting is unnecessary.
				Output	Cluster merging distance information. This information indicates that clusters were merged using distance dis[i − 1] during the i-th merge (i = $1, \cdots,$ nc − 1).

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
11	isw1	I	1	Input	Dissimilarity measure calculation processing switch isw1=11 or 21: Euclidean quadratic distance isw1=12 or 22: Standardized Euclidean quadratic distance isw1=13 or 23: (Generalized) distance of Mahalanobis isw1=14 or 24: Minkowski distance For isw1=11,12,13 or 14, nx=n. For isw1=21,22,23 or 24, ny=n.
12	isw2	I	1	Input	Analysis method selection switch isw2=1: Nearest neighbor method isw2=2: Furtherest neighbor method isw2=3: Group mean method isw2=4: Center of gravity method isw2=5: Median method isw2=6: Ward's method isw2=7: Variable method (input β)
13	iw	I*	nc	Work	Work area
14	w1	$\left\{\begin{matrix}\text{D}* \\ \text{R}*\end{matrix}\right\}$	See Contents	Work	Work area **Size:** When isw1 =11, 12, 13 or 14 the size is ml × nc + ny × (ml + 1) + 2. When isw1 =21, 22, 23 or 24 the size is ml × nc + nx × (ml + 1) + 2.
15	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) $1 < \mathrm{nx} \leq \mathrm{lx}, 1 < \mathrm{ny} \leq \mathrm{ly}$

(b) nc = nx (when isw1 =11, 12, 13 or 14) or ny (when isw1 =21, 22, 23 or 24)

(c) $\mathrm{lnp} \geq \mathrm{nc} - 1$

(d) $\mathrm{ml} = \max(\mathrm{lx}, \mathrm{ly})$

(e) $11 \leq \mathrm{isw1} \leq 14$ or $21 \leq \mathrm{isw2} \leq 24$

(f) $1 \leq \mathrm{isw2} \leq 7$

(g) When isw1 = 14 or 24, $\mathrm{dis}[0] \geq 1.0$

(h) When isw2 = 7, $-1/4 \leq \mathrm{dis}[1] \leq 0$
(Except when entering 1.0 to set the default value)

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
1000	Restriction (h) was not satisfied.	Processing is performed with the default value set.
3000	Restriction (a) or (b) was not satisfied.	Processing is aborted.
3010	Restriction (c) was not satisfied.	
3020	Restriction (d) was not satisfied.	
3030	Restriction (e) or (f) was not satisfied.	
3040	Restriction (g) was not satisfied.	
4000	For the entered observation data, the calculation cannot be performed using the (generalized) distance of Mahalanobis. (see Section 9.1.1)	

(6) **Notes**

(a) When the Euclidean quadratic distance or Minkowski distance is used for the dissimilarity measure calculation and the units of the observation data subjects differ, the data should be standardized in advance so that a suitable positional relationship between the subjects is reflected.

(b) Information indicating which cluster was merged with which cluster during which merge is stored in array ipos. However, the lowest cluster number of the two clusters before merging is assigned to the merged cluster. For example, the cluster formed by merging cluster 3 and cluster 5 is named cluster 3. Where, if nc=nx then m=ny, else if nc=ny then m=nx.

(7) **Example**

(a) Problem

From the following observation data matrix:

$$A = \begin{bmatrix} 0.321 & 119 & 6 & 40 & 6 & 6 & 12 & 29 \\ 0.301 & 112 & 9 & 38 & 4 & 2 & 3 & 31 \\ 0.288 & 133 & 15 & 53 & 4 & 5 & 12 & 30 \\ 0.280 & 112 & 9 & 47 & 4 & 2 & 10 & 24 \\ 0.261 & 109 & 3 & 21 & 1 & 3 & 13 & 21 \\ 0.256 & 107 & 8 & 34 & 4 & 2 & 17 & 38 \\ 0.253 & 95 & 16 & 57 & 4 & 6 & 7 & 37 \\ 0.250 & 100 & 13 & 46 & 4 & 3 & 13 & 37 \\ 0.249 & 92 & 17 & 48 & 6 & 2 & 7 & 23 \\ 0.227 & 88 & 12 & 45 & 4 & 0 & 3 & 33 \end{bmatrix}$$

let the number of rows be the number of subjects and perform a cluster analysis using the group mean method with the dissimilarity measure matrix (standardized Euclidean quadratic distance) as an index.

(b) Input data

Observation data matrix A, lx=11, ly=9, nx=10, ny=8, ml=11, nc=10, lnp=9, isw1=12 and isw2=3.

(c) Main program

```
/*      C interface example for ASL_d6clda */

#include <stdio.h>
#include <stdlib.h>
#include <asl.h>

int main()
{
    double *a;
    int lx, ly;
    int nx, ny;
    int ml;
    int nc;
    int *ipos;
    int lnp;
    double *dis;
    int isw1, isw2;
    int *iw;
    double *w1;
    int ierr;

    int i,j;
    FILE *fp;

    lx=11;
    ly= 9;
    nx=10;
    ny= 8;
    ml=11;
    nc=10;
    lnp=9;
    isw1=12;
    isw2= 3;

    fp = fopen( "d6clda.dat", "r" );
    if( fp == NULL )
    {
        printf( "file open error\n" );
        return -1;
    }

    printf( "    *** ASL_d6clda ***\n" );
    printf( "\n     ** Input **\n\n" );

    a = ( double * )malloc((size_t)( sizeof(double) * (lx*ly) ));
    if( a == NULL )
    {
        printf( "no enough memory for array a\n" );
        return -1;
    }

    w1 = ( double * )malloc((size_t)( sizeof(double) * (ml*nc+ny*(ml+1)+2) ));
    if( w1 == NULL )
```

```
    {
        printf( "no enough memory for array w1\n" );
        return -1;
    }

    dis = ( double * )malloc((size_t)( sizeof(double) * (nc-1) ));
    if( dis == NULL )
    {
        printf( "no enough memory for array dis\n" );
        return -1;
    }

    ipos = ( int * )malloc((size_t)( sizeof(int) * (lnp*2) ));
    if( ipos == NULL )
    {
        printf( "no enough memory for array ipos\n" );
        return -1;
    }

    iw = ( int * )malloc((size_t)( sizeof(int) * nc ));
    if( iw == NULL )
    {
        printf( "no enough memory for array iw\n" );
        return -1;
    }

    printf( "\tisw1=%3d,   isw2=%3d\n", isw1, isw2 );
    printf( "\tlx  =%3d,   ly  =%3d\n", lx, ly );
    printf( "\tnx  =%3d,   ny  =%3d\n", nx, ny );
    printf( "\tml  =%3d,   nc  =%3d\n", ml, nc );
    printf( "\tlnp =%3d\n\n", lnp );

    printf( "\t           A     B     C     D");
    printf( "     E     F     G     H\n");
    printf( "\t------------------------------");
    printf( "--------------------------\n");

    for( i=0 ; i<nx ; i++ )
    {
        printf( "\t%3d | ", i);
        for( j=0 ; j<ny ; j++ )
        {
            fscanf( fp, "%lf", &a[i+lx*j] );
            printf( "%6.3g", a[i+lx*j] );
        }
        printf( "\n" );
    }

    fclose( fp );

    ierr = ASL_d6clda(a, lx, ly, nx, ny, ml, nc,
             ipos, lnp, dis, isw1, isw2, iw, w1);

    printf( "\n     ** Output **\n\n" );
    printf( "\tierr = %6d\n\n", ierr );

    for( i=0 ; i<nc-1 ; i++ )
    {
        printf( "\t%6d --- %6d\t%8.3g\n", ipos[i],ipos[i+lnp],dis[i] );
    }

    free( a );
    free( ipos );
    free( dis );
    free( iw );
    free( w1 );

    return 0;
}
```

(d) Output results

```
*** ASL_d6clda ***

 ** Input **

isw1= 12,   isw2=  3
lx  = 11,   ly  =  9
nx  = 10,   ny  =  8
ml  = 11,   nc  = 10
lnp =  9

           A     B     C     D     E     F     G     H
-------------------------------------------------------------
  0 |  0.321   119     6    40     6     6    12    29
  1 |  0.301   112     9    38     4     2     3    31
  2 |  0.288   133    15    53     4     5    12    30
  3 |   0.28   112     9    47     4     2    10    24
  4 |  0.261   109     3    21     1     3    13    21
  5 |  0.256   107     8    34     4     2    17    38
  6 |  0.253    95    16    57     4     6     7    37
```

```
  7 |   0.25   100    13    46     4     3    13    37
  8 |  0.249    92    17    48     6     2     7    23
  9 |  0.227    88    12    45     4     0     3    33

 ** Output **

ierr =      0

     6 ---      8       3.92
     2 ---      4       4.94
     9 ---     10       8.56
     1 ---      3       10.3
     1 ---      2       10.6
     6 ---      7       11.8
     6 ---      9       13.6
     1 ---      6       15.9
     1 ---      5       25.2
```

Chapter 10

REGRESSION ANALYSIS

10.1 INTRODUCTION

When statistical data consists of multiple variates and an attempt is to be made to predict the value of one of the variates from another of the variates, a **regression analysis** hypothesizes a functional relationship containing several parameters between the two variates and estimates those parameters from observed values. The parameters that are obtained are called the **regression coefficients**. The function obtained by the regression analysis gives a function for approximating the observation data based on some index. This library provides the following functions for performing regression analysis.

- Linear Regression Analysis
- Nonlinear Regression Analysis

10.1.1 Explanation

(1) **Linear Regression Analysis**

Assume that n observed values $(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)$ obey the following linear regression model.

$$y_i = \beta_0 + \beta_1 x_i + \varepsilon_i \quad (i = 1, 2, \cdots, n)$$

Here, ε_i are error terms that independently obey $N(0, \sigma^2)$. Obtaining the estimates b_0 and b_1 of coefficients β_0 and β_1 by using the method of least squares is called a linear regression. b_0 and b_1 are obtained by using the following equations.

$$b_1 = \frac{\sum_{i=1}^{n}(x_i - \mu_x)(y_i - \mu_y)}{\sum_{i=1}^{n}(x_i - \mu_x)^2}$$

$$b_0 = \mu_y - b_1 \mu_x$$

Here, μ_x and μ_y, which represent the means of x_i and y_i, are given by the following equations.

$$\mu_x = \frac{\sum_{i=1}^{n} x_i}{n}, \ \mu_y = \frac{\sum_{i=1}^{n} y_i}{n}$$

The analysis of variance table due to the linear regression is as follows.

	Sum of squares	Degrees of freedom	Unbiased variance	F ratio
Total variation	S_T	$n-1$		
Variation due to regression	S_R	1	$V_R = S_R$	$F = \frac{V_R}{V_E}$
Residual variation	$S_E = S_T - S_R$	$n-2$	$V_E = \frac{S_E}{n-2}$	

Here, S_T and S_R are defined as follows.

$$S_T = \sum_{i=1}^{n}(y_i - \mu_y)^2$$

$$S_R = b_1 \left(\sum_{i=1}^{n}(x_i - \mu_x)(y_i - \mu_y) \right)$$

The estimates ${\varepsilon_0}^2$ and ${\varepsilon_1}^2$ of the variance of the statistics b_0 and b_1, are given by the following equations:

$${\varepsilon_0}^2 = V_E \left(\frac{1}{n} + \frac{{\mu_x}^2}{\sum_{i=1}^{n}(x_i - \mu_x)^2} \right)$$

$$\varepsilon_1{}^2 = \frac{V_E}{\sum_{i=1}^{n}(x_i - \mu_x)^2}$$

and the test quantities t_0, which obeys a t distribution with $n-2$ degrees of freedom based on the null hypothesis $H_0 : \beta_0 = 0$, and t_1, which obeys a t distribution with $n-2$ degrees of freedom based on the null hypothesis $H_0 : \beta_1 = 0$, respectively, are given as follows.

$$t_0 = \frac{b_0}{\varepsilon_0}$$

$$t_1 = \frac{b_1}{\varepsilon_1}$$

(2) **Linear Regression Analysis (Repetitive Data)**

Assume that mi given dependent variable values y_{ij} $(j = 1, 2, \cdots, m_i; i = 1, 2, \cdots, n)$ corresponding to n independent variable values x_i $(i = 1, 2, \cdots, n)$ have been given and obey the following linear regression model.

$$y_{ij} = \beta_0 + \beta_1 x_i + \varepsilon_i \quad (j = 1, 2, \cdots, m_i; i = 1, 2, \cdots, n)$$

Here, ε_i are error terms that independently obey $N(0, \sigma^2)$. The estimates b_0 and b_1 of coefficients β_0 and β_1 are to be obtained by using the method of least squares. b_0 and b_1 are obtained by using the following equations.

$$b_1 = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m_i}(x_i - \mu_x)(y_{ij} - \mu_y)}{\sum_{i=1}^{n} m_i(x_i - \mu_x)^2}$$

$$b_0 = \mu_y - b_1\mu_x$$

Here, μ_x and μ_y, which represent the (weighted) means of x_i and y_{ij}, are given by the following equations.

$$\mu_x = \frac{\sum_{i=1}^{n} m_i x_i}{\sum_{i=1}^{n} m_i}, \ \mu_y = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m_i} y_{ij}}{\sum_{i=1}^{n} m_i}$$

The analysis of variance table due to the linear regression is as follows.

	Sum of squares	Degrees of freedom	Unbiased variance	F ratio
Total variation	S_T	$\sum_{i=1}^{n} m_i - 1$		
Variation due to regression	S_R	1	$V_R = S_R$	$F_R = \dfrac{V_R}{V_E}$
Residual variation	$S_E = S_T - S_R$	$\sum_{i=1}^{n} m_i - 2$	$V_E = \dfrac{S_E}{\sum_{i=1}^{n} m_i - 2}$	
Variation due to high order regression	$S_L = S_B - S_R$	$n - 2$	$V_L = \dfrac{S_L}{n-2}$	$F_L = \dfrac{V_L}{V_W}$
Between-class variation	S_B	$n - 1$	$V_B = \dfrac{S_B}{n-1}$	$F_B = \dfrac{V_B}{V_W}$
Within-class variation	$S_W = S_T - S_B$	$\sum_{i=1}^{n} m_i - n$	$V_W = \dfrac{S_W}{\sum_{i=1}^{n} m_i - n}$	

Here, S_T, S_R and S_B are defined as follows.

$$S_T = \sum_{i=1}^{n}\sum_{j=1}^{m_i}(y_{ij} - \mu_y)^2$$

$$S_R = b_1 \left(\sum_{i=1}^{n}\sum_{j=1}^{m_i}(x_i - \mu_x)(y_{ij} - \mu_y) \right)$$

$$S_B = \sum_{i=1}^{n} m_i \left(\frac{\sum_{j=1}^{m_i} y_{ij}}{m_i} - \mu_y \right)^2$$

The estimates ${\varepsilon_0}^2$ and ${\varepsilon_1}^2$ of the variance of the statistics b_0 and b_1, are given by the following equations:

$${\varepsilon_0}^2 = V_E \left(\frac{1}{\sum_{i=1}^{n} m_i} + \frac{{\mu_x}^2}{\sum_{i=1}^{n} m_i (x_i - \mu_x)^2} \right)$$

$${\varepsilon_1}^2 = \frac{V_E}{\sum_{i=1}^{n} m_i (x_i - \mu_x)^2}$$

and the test quantities t_0, which obeys a t distribution with $\sum_{i=1}^{n} m_i - 2$ degrees of freedom based on the null hypothesis $H_0 : \beta_0 = 0$, and t_1, which obeys a t distribution with $\sum_{i=1}^{n} m_i - 2$ degrees of freedom based on the null hypothesis $H_0 : \beta_1 = 0$, respectively, are given as follows.

$$t_0 = \frac{b_0}{\varepsilon_0}$$

$$t_1 = \frac{b_1}{\varepsilon_1}$$

(3) **Multiple Regression Analysis**

Assume that n independent variable values x_{ki} $(k = 1, 2, \cdots, n; i = 1, 2, \cdots, m)$ for m variables and n given dependent variable values y_k $(k = 1, 2, \cdots, n)$ corresponding to them have been given and obey the following linear regression model.

$$y_k = \beta_0 + \beta_1 x_{k1} + \beta_2 x_{k2} + \cdots + \beta_m x_{km} + \varepsilon_k \quad (k = 1, 2, \cdots, n)$$

Here, ε_k are error terms that independently obey $N(0, \sigma^2)$. Obtaining the estimates b_i $(i = 0, 1, \cdots, m)$ of partial correlation coefficients β_i $(i = 0, 1, \cdots, m)$ by using the method of least squares is called a multiple regression. The b_i $(i = 0, 1, \cdots, m)$ are obtained by solving the following simultaneous linear equations (normal equations) for the $m - 1$ dimensional vector $\boldsymbol{b} = (b_j)$ $(j = 1, 2, \cdots, m)$.

$$C\boldsymbol{b} = \boldsymbol{u}$$

The elements of matrix $C = (c_{ij})$ $(i, j = 1, 2, \cdots, m)$ and vector $\boldsymbol{u} = (u_j)$ $(j = 1, 2, \cdots, m)$ are defined as follows.

$$c_{ij} = \sum_{k=1}^{n} (x_{ki} - \mu_i)(x_{kj} - \mu_j) \ (i, j = 1, 2, \cdots, m)$$

$$u_j = \sum_{k=1}^{n} (x_{ki} - \mu_i)(y_j - \nu) \ (j = 1, 2, \cdots, m)$$

Here, μ_i and ν are given by the following equations.

$$\mu_i = \frac{\sum_{k=1}^{n} x_{ki}}{n} \ (i = 1, 2, \cdots, m)$$

$$\nu = \frac{\sum_{k=1}^{n} y_k}{n}$$

Now, b_0 is obtained as follows:

$$b_0 = \nu - \sum_{i=1}^{m} b_i \mu_i$$

and the analysis of variance table due to the linear regression is as follows.

	Sum of squares	Degrees of freedom	Unbiased variance	F ratio
Total variation	S_T	$n - 1$		
Variation due to regression	S_R	m	$V_R = \dfrac{S_R}{m}$	$F = \dfrac{V_R}{V_E}$
Residual variation	$S_E = S_T - S_R$	$n - m - 1$	$V_E = \dfrac{S_E}{n - m - 1}$	

Here, S_T and S_R are defined as follows.

$$S_T = \sum_{k=1}^{n}(y_k - \nu)^2$$

$$S_R = \sum_{i=1}^{m} b_i \left(\sum_{k=1}^{n}(x_{ki} - \mu_i)(y_i - \nu) \right)$$

Let the elements of the inverse matrix of matrix C be represented by d_{ij} $(i, j = 1, 2, \cdots, m)$. The estimates ${\varepsilon_i}^2$ $(i = 0, 1, \cdots, m)$ of the variance of the statistics b_i $(i = 0, 1, \cdots, m)$ are given by the following equations:

$${\varepsilon_0}^2 = V_E \left(\frac{1}{n} + \sum_{i=1}^{m}\sum_{j=1}^{m} \mu_i \mu_j d_{ij} \right)$$

$${\varepsilon_i}^2 = V_E d_{ii} \ (i = 1, 2, \cdots, m)$$

and the test quantities $t_i (i = 0, 1, \cdots, m)$, which obey t distributions with $n - m - 1$ degrees of freedom based on the null hypotheses $H_0 : \beta_i = 0$ $(i = 0, 1, \cdots, m)$, are given as follows.

$$t_i = \frac{b_i}{\varepsilon_i} \ (i = 0, 1, \cdots, m)$$

(4) **Polynomial Regression Analysis**

Assume that n independent variable values x_k $(k = 1, 2, \cdots, n)$ and n given dependent variable values y_k $(k = 1, 2, \cdots, n)$ corresponding to them have been given and obey the following regression model.

$$y_k = \beta_0 + \beta_1 x_k + \beta_2 {x_k}^2 + \cdots + \beta_m {x_k}^m + \varepsilon_k \quad (k = 1, 2, \cdots, n)$$

Here, ε_k are error terms that independently obey $N(0, \sigma^2)$. The estimates b_i $(i = 0, 1, \cdots, m)$ of partial correlation coefficients β_i $(i = 0, 1, \cdots, m)$ are obtained by using the method of least squares. The b_i $(i = 0, 1, \cdots, m)$ are obtained by solving the following simultaneous linear equations (normal equations) for the $m - 1$ dimensional vector $\boldsymbol{b} = (b_j)$ $(j = 1, 2, \cdots, m)$.

$$C\boldsymbol{b} = \boldsymbol{u}$$

The elements of matrix $C = (c_{ij})$ $(i, j = 1, 2, \cdots, m)$ and vector $\boldsymbol{u} = (u_j)$ $(j = 1, 2, \cdots, m)$ are defined as follows.

$$c_{ij} = \sum_{k=1}^{n}({x_k}^i - \mu_i)({x_k}^j - \mu_j) \ (i, j = 1, 2, \cdots, m)$$

$$u_j = \sum_{k=1}^{n}({x_k}^i - \mu_i)(y_j - \nu) \ (j = 1, 2, \cdots, m)$$

Here, μ_i and ν are given by the following equations.

$$\mu_i = \frac{\sum_{k=1}^{n} {x_k}^i}{n} \ (i = 1, 2, \cdots, m)$$

$$\nu = \frac{\sum_{k=1}^{n} y_k}{n}$$

Now, b_0 is obtained as follows:

$$b_0 = \nu - \sum_{i=1}^{m} b_i \mu_i$$

and the analysis of variance table due to the regression is as follows.

	Sum of squares	Degrees of freedom	Unbiased variance	F ratio
Total variation	S_T	$n-1$		
Variation due to regression	S_R	m	$V_R = \frac{S_R}{m}$	$F = \frac{V_R}{V_E}$
Residual variation	$S_E = S_T - S_R$	$n-m-1$	$V_E = \frac{S_E}{n-m-1}$	

Here, S_T and S_R are defined as follows.

$$S_T = \sum_{k=1}^{n} (y_k - \nu)^2$$

$$S_R = \sum_{i=1}^{m} b_i \left(\sum_{k=1}^{n} ({x_k}^i - \mu_i)(y_i - \nu) \right)$$

Let the elements of the inverse matrix of matrix C be represented by d_{ij} $(i, j = 1, 2, \cdots, m)$. The estimates ${\varepsilon_i}^2$ $(i = 0, 1, \cdots, m)$ of the variance of the statistics b_i $(i = 0, 1, \cdots, m)$ are given by the following equations:

$${\varepsilon_0}^2 = V_E \left(\frac{1}{n} + \sum_{i=1}^{m} \sum_{j=1}^{m} \mu_i \mu_j d_{ij} \right)$$

$${\varepsilon_i}^2 = V_E d_{ii} \ (i = 1, 2, \cdots, m)$$

and the test quantities $t_i (i = 0, 1, \cdots, m)$, which obey t distributions with $n - m - 1$ degrees of freedom based on the null hypotheses $H_0 : \beta_i = 0$ $(i = 0, 1, \cdots, m)$, are given as follows.

$$t_i = \frac{b_i}{\varepsilon_i} \ (i = 0, 1, \cdots, m)$$

Remark: The coefficient matrix of normal equations often is rather ill-behaved, and, in particular, the solution has a tendency to quickly approach a singularity as the number of variates increases. For a polynomial regression, this is serious. Therefore, the calculation should be performed using the double-precision function whenever possible.

(5) **Regression According to an Arbitrary Function**

Assume that n sets of independent variables $\boldsymbol{x}_k = (x_{1k}, x_{2k}, \cdots, x_{mk})$ $(k = 1, 2, \cdots, n)$ consisting of m variables and n given dependent variable values y_k $(k = 1, 2, \cdots, n)$ corresponding to them have been given and obey the following regression model.

$$y_k = f(\boldsymbol{x}_k; \boldsymbol{\beta}) + \varepsilon_k \ (k = 1, 2, \cdots, n)$$

Here, ε_k are error terms that independently obey $N(0, \sigma^2)$. The estimates $\boldsymbol{b} = (b_1, b_2, \cdots, b_\ell)$ of the ℓ regression model parameters $\boldsymbol{\beta} = (\beta_1, \beta_2, \cdots, \beta_\ell)$ is obtained according to the nonlinear method of least squares, with $\boldsymbol{\beta}$ minimizing the residual variation

$$S_e = \sum_{k=1}^{n} (y_k - f(x_k; \boldsymbol{\beta}))^2,$$

and the analysis of variance table due to the regression is as follows.

	Sum of squares	Degrees of freedom	Unbiased variance
Total variation	S_T	n	
Residual variation	S_E	$n - \ell$	$V_E = \dfrac{S_E}{n - \ell}$

Here, S_T and S_E are defined as follows.

$$S_T = \sum_{k=1}^{n} y_k^2$$

$$S_E = \sum_{i=1}^{n} (y_k - f(\boldsymbol{x}_k; \boldsymbol{b}))^2$$

In addition, the asymptotic variance-covariance matrix $V = (V_{ij})$ $(i, j = 1, \cdots, \ell)$ and standard deviation estimates ε_i $(i = 1, \cdots, \ell)$ of the statistics b_i $(i = 1, \cdots, \ell)$ are defined by the following equations.

$$V = S_E (J^T J)^{-1}$$

$$\varepsilon_i = \sqrt{V_{ii}} \ (i = 1, 2, \cdots, \ell)$$

Here, J is the Jacobian matrix for which the (i, j)-th element is defined as follows.

$$J_{ij} = \left. \frac{\partial f(\boldsymbol{x}_i; \boldsymbol{\beta})}{\partial \beta_j} \right|_{\boldsymbol{\beta} = \boldsymbol{b}} \qquad (i = 1, \cdots, n; j = 1, \cdots, \ell)$$

10.1.2 Reference Bibliography

(1) Draper, N. R. and Smith, H. , "Applied regression analysis", John Wiley & Sons, New York (1966)

10.2 LINEAR REGRESSION ANALYSIS

10.2.1 ASL_dnlnrg, ASL_rnlnrg
Linear Regression Analysis

(1) **Function**

Assume that n observed values $(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)$ obey the following linear regression model.

$$y_i = \beta_0 + \beta_1 x_i + \varepsilon_i \quad (i = 1, 2, \cdots, n)$$

Here, ε_i are error terms that independently obey $N(0, \sigma^2)$.

The ASL_dnlnrg or ASL_rnlnrg obtains:

(a) the estimates b_0 and b_1 of coefficients β_0 and β_1

(b) the statistics for creating the following analysis of variance table due to the linear regression:

	Sum of squares	Degrees of freedom	Unbiased variance	F ratio
Total variation	S_T	$n-1$		
Variation due to regression	S_R	1	$V_R = S_R$	$F = \dfrac{V_R}{V_E}$
Residual variation	$S_E = S_T - S_R$	$n-2$	$V_E = \dfrac{S_E}{n-2}$	

(c) the multiple correlation coefficient R

$$R = \sqrt{\frac{S_R}{S_T}}$$

(d) the standard deviation estimates ε_0 and ε_1 of the statistics b_0 and b_1

(e) the test quantity t_0, which obeys a t distribution with $n-2$ degrees of freedom based on the null hypothesis $H_0 : \beta_0 = 0$

(f) the test quantity t_1, which obeys a t distribution with $n-2$ degrees of freedom based on the null hypothesis $H_0 : \beta_1 = 0$.

(2) **Usage**

Double precision:

```
ierr = ASL_dnlnrg  (x, y, n, b0, b1, r, stat);
```

Single precision:

```
ierr = ASL_rnlnrg  (x, y, n, b0, b1, r, stat);
```

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	x	{D∗ / R∗}	n	Input	Independent variate observation data x_i
2	y	{D∗ / R∗}	n	Input	Dependent variate observation data y_i
3	n	I	1	Input	Number of observed values n
4	b0	{D∗ / R∗}	3	Output	b0[0]: β_0 estimate b_0 b0[1]: β_0 standard deviation estimate ε_0 b0[2]: β_0 test quantity t_0
5	b1	{D∗ / R∗}	3	Output	b1[0]: β_1 estimate b_1 b1[1]: β_1 standard deviation estimate ε_1 b1[2]: β_1 test quantity t_1
6	r	{D∗ / R∗}	2	Output	r[0]: Multiple correlation coefficient R r[1]: Contribution ratio R^2
7	stat	{D∗ / R∗}	21	Output	Basic statistics of calculation result (See Note (a))
8	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) n ≥ 3

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
1000	There was no difference between independent variates.	0.0 is set for b0, b1, r, and stat[0] to stat[8].
1010	The residual is 0.0. (stat[2]=0.0)	The positive maximum value is set for stat[8], b0[2], and b1[2].
3000	Restriction (a) was not satisfied.	Processing is aborted.

(6) **Notes**

(a) The following kind of array is stored as a one-dimensional vector in array stat.

1:	S_T	Total variation
2:	S_R	Variation due to regression
3:	S_E	Residual variation
4:	$f_T = n - 1$	Degrees of freedom of total variation
5:	$f_R = 1$	Degrees of freedom of variation due to regression
6:	$f_E = n - 2$	Degrees of freedom of residual variation
7:	V_R	Unbiased variance of variation due to regression
8:	V_E	Unbiased variance of residual variation
9:	F	F ratio
10:	$\mu_x = \frac{\sum_{i=1}^{n} x_i}{n}$	Mean of x_i
11:	$\mu_y = \frac{\sum_{i=1}^{n} y_i}{n}$	Mean of y_i
12:	$\sigma_x = \sqrt{\frac{\sum_{i=1}^{n} (x_i - \mu_x)^2}{n-1}}$	Standard deviation of x_i
13:	$\sigma_y = \sqrt{\frac{\sum_{i=1}^{n} (y_i - \mu_y)^2}{n-1}}$	Standard deviation of y_i
14:	$s_x = \sum_{i=1}^{n} x_i$	Sum of x_i
15:	$s_y = \sum_{i=1}^{n} y_i$	Sum of y_i
16:	$s_{xx} = \sum_{i=1}^{n} {x_i}^2$	Sum of squares of x_i
17:	$s_{yy} = \sum_{i=1}^{n} {y_i}^2$	Sum of squares of y_i
18:	$s_{xy} = \sum_{i=1}^{n} x_i y_i$	Sum of products of x_i and y_i
19:	$S_{xx} = \sum_{i=1}^{n} (x_i - \mu_x)^2$	Sum of squares of deviations of x_i
20:	$S_{yy} = \sum_{i=1}^{n} (y_i - \mu_y)^2$	Sum of squares of deviations of y_i
21:	$S_{xy} = \sum_{i=1}^{n} (x_i - \mu_x)(y_i - \mu_y)$	Sum of products of deviations of x_i and y_i

(7) **Example**

(a) Problem

Obtain the regression coefficient values, multiple correlation coefficient value, contribution ratio, and basic statistics according to a regression analysis from the following observation data.

x[0]=1.0	y[0]=-2.0
x[1]=2.0	y[1]=7.0
x[2]=3.0	y[2]=34.0
x[3]=4.0	y[3]=91.0
x[4]=5.0	y[4]=190.0
x[5]=6.0	y[5]=343.0
x[6]=7.0	y[6]=562.0
x[7]=8.0	y[7]=859.0
x[8]=9.0	y[8]=1246.0
x[9]=10.0	y[9]=1735.0
x[10]=11.0	y[10]=2338.0

(b) Input data

Observation data x, y and n=11.

(c) Main program

```
/*      C interface example for ASL_dnlnrg */

#include <stdio.h>
#include <stdlib.h>
#include <asl.h>

int main()
{
    double *x, *y;
    int n;
    double b0[3], b1[3], r[2], stat[21];
    int ierr;

    int i;

    FILE *fp;


    fp = fopen( "dnlnrg.dat", "r" );
    if( fp == NULL )
    {
        printf( "file open error\n" );
        return -1;
    }

    printf( "    *** ASL_dnlnrg ***\n" );
    printf( "\n     ** Input **\n\n" );

    fscanf( fp, "%d", &n );

    printf( "\tn=%3d\n\n", n );

    x = ( double * )malloc((size_t)( sizeof(double) * (n) ));
    if( x == NULL )
    {
        printf( "no enough memory for array x\n" );
        return -1;
    }

    y = ( double * )malloc((size_t)( sizeof(double) * (n) ));
    if( y == NULL )
    {
        printf( "no enough memory for array y\n" );
        return -1;
    }

    for( i=0 ; i<n ; i++ )
    {
        fscanf( fp, "%lf", &x[i] );
    }

    for( i=0 ; i<n ; i++ )
    {
```

```
        fscanf( fp, "%lf", &y[i] );
    }

    printf( "\tData of x and y\n");
    for( i=0 ; i<n ; i++ )
    {
        printf( "\t%8.3g  %8.4g\n",x[i], y[i] );
    }

    fclose( fp );

    ierr = ASL_dnlnrg(x, y, n, b0, b1, r, stat);

    printf( "\n     ** Output **\n\n" );
    printf( "\tierr = %6d\n\n", ierr );

    printf( "\tRegression coefficient\n" );
    printf( "\t             R.C.    S.E.    T-V.\n" );
    printf( "\tx        " );
    for( i=0 ; i<3 ; i++ )
    {
        printf( "%8.3g", b1[i] );
    }
    printf( "\n" );
    printf( "\tconstant " );
    for( i=0 ; i<3 ; i++ )
    {
        printf( "%8.3g", b0[i] );
    }
    printf( "\n\n" );

    printf( "\tMultiple correlation coefficient=%8.3g\n", r[0] );
    printf( "\t              Contribution ratio=%8.3g\n\n", r[1] );

    printf( "\tAnalysis of variance table\n" );
    printf( "\t                   S.S     D.F          U.V      F-R\n" );
    printf( "\tTotal      %11.3e  %6.0f\n", stat[0],stat[3] );
    printf( "\tRegression %11.3e  %6.0f  %11.3e %8.3g\n",
        stat[1],stat[4],stat[6],stat[8] );
    printf( "\tDeviation  %11.3e  %6.0f  %11.3e\n\n",
        stat[2],stat[5],stat[7] );

    printf( "\tStatistics\n" );

    printf( "\tMean of x                  : %11.3e\n", stat[9] );
    printf( "\tMean of y                  : %11.3e\n", stat[10] );
    printf( "\tStandard deviation of x    : %11.3e\n", stat[11] );
    printf( "\tStandard deviation of y    : %11.3e\n", stat[12] );
    printf( "\tSum of x                   : %11.3e\n", stat[13] );
    printf( "\tSum of y                   : %11.3e\n", stat[14] );
    printf( "\tSum of squares of x        : %11.3e\n", stat[15] );
    printf( "\tSum of squares of y        : %11.3e\n", stat[16] );
    printf( "\tSum of products of x and y : %11.3e\n", stat[17] );
    printf( "\tSum of squares of \n" );
    printf( "\t    deviations of x        : %11.3e\n", stat[18] );
    printf( "\tSum of squares of \n" );
    printf( "\t    deviations of y        : %11.3e\n", stat[19] );
    printf( "\tSum of products of \n" );
    printf( "\t    deviations of x and y  : %11.3e\n", stat[20] );

    free( x );
    free( y );

    return 0;
}
```

(d) Output results

```
*** ASL_dnlnrg ***

 ** Input **

n= 11

Data of x and y
       1        -2
       2         7
       3        34
       4        91
       5       190
       6       343
       7       562
       8       859
       9      1246
      10      1735
      11      2338

 ** Output **
```

```
ierr =      0

Regression coefficient
             R.C.    S.E.    T-V.
x             220    31.1    7.06
constant     -645     211   -3.05

Multiple correlation coefficient=    0.92
              Contribution ratio=   0.847

Analysis of variance table
                   S.S     D.F          U.V      F-R
Total        6.264e+06      10
Regression   5.305e+06       1    5.305e+06     49.8
Deviation    9.591e+05       0    1.066e+05

Statistics
Mean of x                  :   6.000e+00
Mean of y                  :   6.730e+02
Standard deviation of x    :   3.317e+00
Standard deviation of y    :   7.914e+02
Sum of x                   :   6.600e+01
Sum of y                   :   7.403e+03
Sum of squares of x        :   5.060e+02
Sum of squares of y        :   1.125e+07
Sum of products of x and y :   6.857e+04
Sum of squares of
    deviations of x        :   1.100e+02
Sum of squares of
    deviations of y        :   6.264e+06
Sum of products of
    deviations of x and y  :   2.416e+04
```

10.2.2 ASL_dnlnrr, ASL_rnlnrr
Linear Regression Analysis (Repetitive Data)

(1) **Function**

Assume that m_i given dependent variable values y_{ij} $(j = 1, 2, \cdots, m_i; i = 1, 2, \cdots, n)$ corresponding to n independent variable values x_i $(i = 1, 2, \cdots, n)$ have been given and obey the following linear regression model.

$$y_{ij} = \beta_0 + \beta_1 x_i + \varepsilon_i \quad (j = 1, 2, \cdots, m_i; i = 1, 2, \cdots, n)$$

Here, ε_i are error terms that independently obey $N(0, \sigma^2)$.

The ASL_dnlnrr or ASL_rnlnrr obtains:

(a) the estimates b_0 and b_1 of coefficients β_0 and β_1

(b) the statistics for creating the following analysis of variance table due to the linear regression.

	Sum of squares	Degrees of freedom	Unbiased variance	F ratio
Total variation	S_T	$\sum_{i=1}^{n} m_i - 1$		
Variation due to regression	S_R	1	$V_R = S_R$	$F_R = \dfrac{V_R}{V_E}$
Residual variation	$S_E = S_T - S_R$	$\sum_{i=1}^{n} m_i - 2$	$V_E = \dfrac{S_E}{\sum_{i=1}^{n} m_i - 2}$	
Variation due to high order regression	$S_L = S_B - S_R$	$n - 2$	$V_L = \dfrac{S_L}{n-2}$	$F_L = \dfrac{V_L}{V_W}$
Between-class variation	S_B	$n - 1$	$V_B = \dfrac{S_B}{n-1}$	$F_B = \dfrac{V_B}{V_W}$
Within-class variation	$S_W = S_T - S_B$	$\sum_{i=1}^{n} m_i - n$	$V_W = \dfrac{S_W}{\sum_{i=1}^{n} m_i - n}$	

(c) the multiple correlation coefficient R

$$R = \sqrt{\frac{S_R}{S_T}}$$

(d) the standard deviation estimates ε_0 and ε_1 of the statistics b_0 and b_1

(e) the test quantity t_0, which obeys a t distribution with $\sum_{i=1}^{n} m_i - 2$ degrees of freedom based on the null hypothesis $H_0 : \beta_0 = 0$

(f) the test quantity t_1, which obeys a t distribution with $\sum_{i=1}^{n} m_i - 2$ degrees of freedom based on the null hypothesis $H_0 : \beta_1 = 0$

(2) **Usage**

Double precision:

```
ierr = ASL_dnlnrr  (x, n, ny, y, my, m, b0, b1, r, stat);
```

Single precision:

```
ierr = ASL_rnlnrr  (x, n, ny, y, my, m, b0, b1, r, stat);
```

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	x	{D∗ / R∗}	n	Input	Independent variate observation data x_i
2	n	I	1	Input	Number of independent variate observed values n
3	ny	I*	n	Input	Number of repetitions of each dependent variate for an independent variate m_i
4	y	{D∗ / R∗}	my × m	Input	Dependent variate observation data y_{ij} (See Note (a))
5	my	I	1	Input	Adjustable dimension of array y
6	m	I	1	Input	Maximum number of repetitions of dependent variate observed values $\max(m_i)$ (See Note (c))
7	b0	{D∗ / R∗}	3	Output	b0[0]: β_0 estimate b_0 b0[1]: β_0 standard deviation estimate ε_0 b0[2]: β_0 test quantity t_0
8	b1	{D∗ / R∗}	3	Output	b1[0]: β_1 estimate b_1 b1[1]: β_1 standard deviation estimate ε_1 b1[2]: β_1 test quantity t_1
9	r	{D∗ / R∗}	2	Output	r[0]: Multiple correlation coefficient R r[1]: Contribution ratio R^2
10	stat	{D∗ / R∗}	33	Output	Basic statistics of calculation result (See Note (b))
11	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) $3 \leq \mathrm{n} \leq \mathrm{my}$

(b) $1 \leq \mathrm{ny[i-1]} \leq \mathrm{m} \quad (\mathrm{i} = 1, 2, \cdots, \mathrm{n})$

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
1000	There was no difference between independent variates.	0.0 is set for b0, b1, r, and stat[0] to stat[19].
3000	Restriction (a) was not satisfied.	Processing is aborted.
3010	Restriction (b) was not satisfied.	

(6) **Notes**

(a) Observation data $y_{i,j}$ is stored in array y as the following kind of real matrix (two-dimensional array type) data (See Appendix A).

$$\begin{bmatrix} y_{1,1} & y_{1,2} & \cdots & y_{1,m_1} & * & * \\ y_{2,1} & y_{2,2} & \cdots & \cdots & y_{2,m_2} & * \\ \vdots & & & \vdots & * & * \\ y_{i,1} & & \cdots & \cdots & \cdots & y_{i,m_i} \\ \vdots & & & \vdots & * & * \\ y_{n,1} & \cdots & y_{n,m_n} & * & * & * \end{bmatrix}$$

Remark: The asterisks ($*$) indicate arbitrary values.

(b) The following kind of array is stored as a one-dimensional vector in array stat.

1:	S_T	Total variation
2:	S_R	Variation due to regression
3:	S_E	Residual variation
4:	$f_T = \sum_{i=1}^{n} m_i - 1$	Degrees of freedom of total variation
5:	$f_R = 1$	Degrees of freedom of variation due to regression
6:	$f_E = \sum_{i=1}^{n} m_i - 2$	Degrees of freedom of residual variation
7:	V_R	Unbiased variance of variation due to regression
8:	V_E	Unbiased variance of residual variation
9:	F_T	F ratio of variation due to regression
10:	S_L	Variation due to high-order regression
11:	S_B	Between-class variation
12:	S_W	Within-class variation
13:	$f_L = n - 2$	Degrees of freedom due to high-order regression
14:	$f_B = n - 1$	Degrees of freedom of between-class variation
15:	$f_W = \sum_{i=1}^{n} m_i - n$	Degrees of freedom of within-class variation

16:	V_L	Unbiased variance of variation due to high-order regression
17:	V_B	Unbiased variance of between-class variation
18:	V_W	Unbiased variance of within-class variation
19:	F_L	F ratio of variation due to high-order regression
20:	F_B	F ratio of between-class variation
21:	$\mu_x = \frac{\sum_{i=1}^{n} m_i x_i}{\sum_{i=1}^{n} m_i}$	Weighted mean of x_i
22:	$\mu_y = \frac{\sum_{i=1}^{n} \sum_{j=1}^{m_i} y_{ij}}{\sum_{i=1}^{n} m_i}$	Grand mean of y_{ij}
23:	$\sigma_x = \sqrt{\frac{\sum_{i=1}^{n} m_i (x_i - \mu_x)^2}{\sum_{i=1}^{n} m_i - 1}}$	Weighted standard deviation of x_i
24:	$\sigma_y = \sqrt{\frac{\sum_{i=1}^{n} \sum_{j=1}^{m_i} (y_{ij} - \mu_y)^2}{\sum_{i=1}^{n} m_i - 1}}$	Standard deviation of y_{ij}
25:	$s_x = \sum_{i=1}^{n} m_i x_i$	Weighted sum of x_i
26:	$s_y = \sum_{i=1}^{n} \sum_{j=1}^{m_i} y_{ij}$	Sum of y_{ij}
27:	$s_{xx} = \sum_{i=1}^{n} m_i {x_i}^2$	Weighted sum of squares of x_i
28:	$s_{yy} = \sum_{i=1}^{n} \sum_{j=1}^{m_i} {y_{ij}}^2$	Sum of squares of y_{ij}
29:	$s_{xy} = \sum_{i=1}^{n} \sum_{j=1}^{m_i} x_i y_{ij}$	Sum of products of x_i and y_{ij}
30:	$S_{xx} = \sum_{i=1}^{n} m_i (x_i - \mu_x)^2$	Weighted sum of squares of deviations of x_i
31:	$S_{yy} = \sum_{i=1}^{n} \sum_{j=1}^{m_i} (y_{ij} - \mu_y)^2$	Sum of squares of deviations of y_{ij}
32:	$S_{xy} = \sum_{i=1}^{n} \sum_{j=1}^{m_i} (x_i - \mu_x)(y_{ij} - \mu_y)$	Sum of products of deviations of x_i and y_{ij}
33:	$\sum_{i=1}^{n} m_i$	Total number of y_{ij}

(c) When m=1, stat[9] to stat[19] are not calculated.

(7) **Example**

(a) Problem

Obtain the regression coefficient values, multiple correlation coefficient value, contribution ratio, and basic statistics according to a regression analysis from the following observation data:

$$(x_i) = \begin{bmatrix} 55.0 \\ 65.0 \\ 75.0 \\ 80.0 \\ 85.0 \end{bmatrix}, \quad (y_{i,j}) = \begin{bmatrix} 26.0 & 29.0 & 32.0 & 0.0 \\ 32.0 & 35.0 & 0.0 & 0.0 \\ 36.0 & 35.0 & 31.0 & 34.0 \\ 33.0 & 0.0 & 0.0 & 0.0 \\ 37.0 & 35.0 & 0.0 & 0.0 \end{bmatrix}$$

and numbers of repetitions:

$$(m_i) = \begin{bmatrix} 3 \\ 2 \\ 4 \\ 1 \\ 2 \end{bmatrix}$$

(b) Input data

Observation data x and y, numbers of repetitions ny, n=5, my=5 and m=4.

(c) Main program

```
/*      C interface example for ASL_dnlnrr */

#include <stdio.h>
#include <stdlib.h>
#include <asl.h>

int main()
{
    double *x;
    int n;
    int *ny;
    double *y;
    int my;
    int m;
    double b0[3], b1[3], r[2], stat[33];
    int ierr;

    int i,j;

    FILE *fp;


    fp = fopen( "dnlnrr.dat", "r" );
    if( fp == NULL )
    {
        printf( "file open error\n" );
        return -1;
    }

    printf( "    *** ASL_dnlnrr ***\n" );
    printf( "\n     ** Input **\n\n" );

    fscanf( fp, "%d", &my );
    fscanf( fp, "%d", &n );
    fscanf( fp, "%d", &m );

    printf( "\tmy=%3d\n", my );
    printf( "\t n=%3d\n", n );
    printf( "\t m=%3d\n\n", m );

    x = ( double * )malloc((size_t)( sizeof(double) * (n) ));
    if( x == NULL )
    {
        printf( "no enough memory for array x\n" );
        return -1;
    }

    ny = ( int * )malloc((size_t)( sizeof(int) * (n) ));
    if( ny == NULL )
    {
        printf( "no enough memory for array ny\n" );
```

```
    return -1;
}

y = ( double * )malloc((size_t)( sizeof(double) * (my*m) ));
if( y == NULL )
{
    printf( "no enough memory for array y\n" );
    return -1;
}

for( i=0 ; i<n ; i++ )
{
    fscanf( fp, "%d", &ny[i] );
}

for( i=0 ; i<n ; i++ )
{
    fscanf( fp, "%lf", &x[i] );
}

for( j=0 ; j<m ; j++ )
{
    for( i=0 ; i<my ; i++ )
    {
        fscanf( fp, "%lf", &y[i+my*j] );
    }
}

printf( "\tData of x and y\n");
for( i=0 ; i<n ; i++ )
{
    printf( "\t%8.3g  | %4d | ",x[i], ny[i] );
    for( j=0 ; j<ny[i] ; j++ )
    {
        printf( "%8.3g",y[i+my*j] );
    }
    printf( "\n" );
}

fclose( fp );

ierr = ASL_dnlnrr(x, n, ny, y, my, m, b0, b1, r, stat);

printf( "\n     ** Output **\n\n" );
printf( "\tierr = %6d\n\n", ierr );

printf( "\tRegression coefficient\n" );
printf( "\t             R.C.    S.E.    T-V.\n" );
printf( "\tx        " );
for( i=0 ; i<3 ; i++ )
{
    printf( "%8.3g", b1[i] );
}
printf( "\n" );
printf( "\tconstant " );
for( i=0 ; i<3 ; i++ )
{
    printf( "%8.3g", b0[i] );
}
printf( "\n\n" );

printf( "\tMultiple correlation coefficient=%8.3g\n", r[0] );
printf( "\t              Contribution ratio=%8.3g\n\n", r[1] );

printf( "\tAnalysis of variance table\n" );
printf( "\t                     S.S     D.F       U.V      F-R\n" );
printf( "\tTotal           %8.3g  %6.0f\n", stat[0],stat[3] );
printf( "\tRegression      %8.3g  %6.0f  %8.3g %8.3g\n",
    stat[1],stat[4],stat[6],stat[8] );
printf( "\tDeviation       %8.3g  %6.0f  %8.3g\n",
    stat[2],stat[5],stat[7] );
printf( "\tLack of fitness %8.3g  %6.0f  %8.3g %8.3g\n",
    stat[9],stat[12],stat[15],stat[18] );
printf( "\tBetween-classes %8.3g  %6.0f  %8.3g %8.3g\n",
    stat[10],stat[13],stat[16],stat[19] );
printf( "\tWithin-class    %8.3g  %6.0f  %8.3g\n\n",
    stat[11],stat[14],stat[17] );

printf( "\tStatistics\n" );

printf( "\tWeighted mean of x           : %9.3g\n", stat[20] );
printf( "\tGrand mean of y              : %9.3g\n", stat[21] );
printf( "\tWeighted standard \n" );
printf( "\t    deviation of x           : %9.3g\n", stat[22] );
printf( "\tStandard deviation of y      : %9.3g\n", stat[23] );
printf( "\tWeighted sum of x            : %9.3g\n", stat[24] );
printf( "\tSum of y                     : %9.3g\n", stat[25] );
```

```
    printf( "\tWeighted sum of squares of x : %9.4g\n", stat[26] );
    printf( "\tSum of squares of y          : %9.4g\n", stat[27] );
    printf( "\tSum of products of x and y   : %9.4g\n", stat[28] );
    printf( "\tWeighted sum of squares of \n" );
    printf( "\t    deviations of x          : %9.4g\n", stat[29] );
    printf( "\tSum of squares of \n" );
    printf( "\t    deviations of y          : %9.3g\n", stat[30] );
    printf( "\tSum of products of \n" );
    printf( "\t    deviations of x and y    : %9.3g\n", stat[31] );
    printf( "\tTotal number of y            : %9.3g\n", stat[32] );

    free( x );
    free( ny );
    free( y );

    return 0;
}
```

(d) Output results

```
*** ASL_dnlnrr ***

 ** Input **

my=  5
 n=  5
 m=  4

Data of x and y
      55  |    3 |       26      29      32
      65  |    2 |       32      35
      75  |    4 |       36      35      31      34
      80  |    1 |       33
      85  |    2 |       37      35

 ** Output **

ierr =      0

Regression coefficient
             R.C.    S.E.    T-V.
x           0.208  0.0601    3.46
constant     18.3    4.28    4.27

Multiple correlation coefficient=   0.738
              Contribution ratio=   0.545

Analysis of variance table
                     S.S     D.F       U.V      F-R
Total                109      11
Regression          59.3       1      59.3       12
Deviation           49.6      10      4.96
Lack of fitness     11.1       3      3.69    0.672
Between-classes     70.4       4      17.6      3.2
Within-class        38.5       7       5.5

Statistics
Weighted mean of x           :      70.4
Grand mean of y              :      32.9
Weighted standard
    deviation of x           :      11.2
Standard deviation of y      :      3.15
Weighted sum of x            :       845
Sum of y                     :       395
Weighted sum of squares of x : 6.088e+04
Sum of squares of y          : 1.311e+04
Sum of products of x and y   :  2.81e+04
Weighted sum of squares of
    deviations of x          :      1373
Sum of squares of
    deviations of y          :       109
Sum of products of
    deviations of x and y    :       285
Total number of y            :        12
```

10.2.3 ASL_dnlnma, ASL_rnlnma Multiple Regression Analysis

(1) **Function**

Assume that n independent variable values x_{ki} $(k = 1, 2, \cdots, n; i = 1, 2, \cdots, m)$ for m variables and n given dependent variable values y_k $(k = 1, 2, \cdots, n)$ corresponding to them have been given and obey the following linear regression model.

$$y_k = \beta_0 + \beta_1 x_{k1} + \beta_2 x_{k2} + \cdots + \beta_m x_{km} + \varepsilon_k \quad (k = 1, 2, \cdots, n)$$

Here, ε_k are error terms that independently obey $N(0, \sigma^2)$.

The ASL_dnlnma or ASL_rnlnma obtains:

(a) the estimates b_i $(i = 0, 1, \cdots, m)$ of partial correlation coefficients β_i $(i = 0, 1, \cdots, m)$

(b) the statistics for creating the following analysis of variance table due to the linear regression.

	Sum of squares	Degrees of freedom	Unbiased variance	F ratio
Total variation	S_T	$n-1$		
Variation due to regression	S_R	m	$V_R = \dfrac{S_R}{m}$	$F = \dfrac{V_R}{V_E}$
Residual variation	$S_E = S_T - S_R$	$n-m-1$	$V_E = \dfrac{S_E}{n-m-1}$	

(c) the multiple correlation coefficient R

$$R = \sqrt{\frac{S_R}{S_T}}$$

(d) the degree of freedom adjusted correlation coefficient R^*

$$R^* = \sqrt{1 - \frac{(n-1)S_E}{(n-m-1)S_T}}$$

(e) the standard deviation estimates ε_i $(i = 0, 1, \cdots, m)$ of the statistics b_i $(i = 0, 1, \cdots, m)$

(f) the test quantities $t_i (i = 0, 1, \cdots, m)$, which obey t distributions with $n - m - 1$ degrees of freedom based on the null hypotheses $H_0 : \beta_i = 0$ $(i = 0, 1, \cdots, m)$

(2) **Usage**

Double precision:

```
ierr = ASL_dnlnma  (x, mx, m, n, b, r, v, stat, iw1, w1);
```

Single precision:

```
ierr = ASL_rnlnma  (x, mx, m, n, b, r, v, stat, iw1, w1);
```

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	x	{D∗ / R∗}	mx, m+1	Input	Independent variate observation data x_{ki} and dependent variate observation data y_k (See Note (a))
2	mx	I	1	Input	Adjustable dimension of array x
3	m	I	1	Input	Number of independent variates m
4	n	I	1	Input	Number of observed values n
5	b	{D∗ / R∗}	$3\times(m+1)$	Output	b[i]: β_i estimate b_i b[i+m+1]: β_i standard deviation estimate ε_i b[i+2×(m+1)]: β_i test quantity t value t_i ($i = 0, 1, \cdots, m$)
6	r	{D∗ / R∗}	3	Output	r[0]: Multiple correlation coefficient R r[1]: Contribution ratio R^2 r[2]: Degree of freedom adjusted correlation coefficient R^*
7	v	{D∗ / R∗}	9	Output	Statistics for creating analysis of variance table (See Note (b))
8	stat	{D∗ / R∗}	5×(m+1)	Output	Basic statistics of calculation result (See Note (c))
9	iw1	I*	m+1	Work	Work area
10	w1	{D∗ / R∗}	See Contents	Work	Work area **Size:** $(m+1)\times(m+3)$
11	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) $3 \le n \le mx$

(b) $1 \le m < n-1$

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
1000	Total sum of squares $\leq$ Regression sum of squares.	0.0 is set for b_i and t_i.
2000	Residual unbiased variance > Total unbiased variance.	0.0 is set for R^*.
3000	Restriction (a) was not satisfied.	Processing is aborted.
3010	Restriction (b) was not satisfied.	
4000	The inverse matrix could not be obtained.	

(6) **Notes**

(a) Observation data x_{ki} $(k = 1, 2, \cdots, n; i = 1, 2, \cdots, m)$ and y_k $(k = 1, 2, \cdots, n)$ are stored in array x as the following kind of real matrix (two-dimensional array type) data (See Appendix A).

$$\begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1,m} & y_1 \\ x_{21} & \ddots & & \vdots & y_2 \\ \vdots & & \ddots & \vdots & \vdots \\ x_{n1} & \cdots & \cdots & x_{nm} & y_n \end{bmatrix}$$

(b) The following kind of a one-dimensional vector is stored in array v.

1:	S_T	Total variation
2:	S_R	Variation due to regression
3:	S_E	Residual variation
4:	$f_T = n - 1$	Degrees of freedom of total variation
5:	$f_R = m$	Degrees of freedom of variation due to regression
6:	$f_E = n - m - 1$	Degrees of freedom of residual variation
7:	V_R	Unbiased variance of variation due to regression
8:	V_E	Unbiased variance of residual variation
9:	F	F ratio

(c) The following kind of real matrix (two-dimensional array type) is stored in array stat (See Appendix A).

$$\begin{bmatrix} \sum_{k=1}^{n} x_{k1} & \sum_{k=1}^{n} x_{k2} & \cdots & \sum_{k=1}^{n} x_{k,m} & \sum_{k=1}^{n} y_k \\ \mu_1 & \mu_2 & \cdots & \mu_m & \nu \\ \sum_{k=1}^{n}(x_{k1}-\mu_1)^2 & \sum_{k=1}^{n}(x_{k2}-\mu_2)^2 & \cdots & \sum_{k=1}^{n}(x_{km}-\mu_m)^2 & \sum_{k=1}^{n}(y_k-\nu)^2 \\ \dfrac{\sum_{k=1}^{n}(x_{k1}-\mu_1)^2}{n-1} & \dfrac{\sum_{k=1}^{n}(x_{k2}-\mu_2)^2}{n-1} & \cdots & \dfrac{\sum_{k=1}^{n}(x_{km}-\mu_m)^2}{n-1} & \dfrac{\sum_{k=1}^{n}(y_k-\nu)^2}{n-1} \\ \sqrt{\dfrac{\sum_{k=1}^{n}(x_{k1}-\mu_1)^2}{n-1}} & \sqrt{\dfrac{\sum_{k=1}^{n}(x_{k2}-\mu_2)^2}{n-1}} & \cdots & \sqrt{\dfrac{\sum_{k=1}^{n}(x_{km}-\mu_m)^2}{n-1}} & \sqrt{\dfrac{\sum_{k=1}^{n}(y_k-\nu)^2}{n-1}} \end{bmatrix}$$

(7) **Example**

(a) Problem

Obtain the regression coefficient values, multiple correlation coefficient value, contribution ratio, analysis of variance table, and basic statistics according to a regression analysis from the following observation data.

$$(x_{ki}|y_k) = \left[\begin{array}{rrr|r} 2.0 & 1.0 & 3.0 & -1.0 \\ 3.0 & 3.0 & -1.0 & 0.0 \\ 4.0 & -2.0 & 0.0 & 2.0 \\ 1.0 & 1.0 & -2.0 & 2.0 \\ -1.0 & 3.0 & -1.0 & 3.0 \end{array}\right]$$

(b) Input data

Observation data x, mx=5, n=5 and m=3.

(c) Main program

```
/*      C interface example for ASL_dnlnma */

#include <stdio.h>
#include <stdlib.h>
#include <asl.h>

int main()
{
    double *x;
    int mx, m, n;
    double *b, r[3], v[9], *stat;
    int *iw1;
    double *w1;
    int ierr;

    int i,j;

    FILE *fp;


    fp = fopen( "dnlnma.dat", "r" );
    if( fp == NULL )
    {
        printf( "file open error\n" );
        return -1;
    }

    printf( "    *** ASL_dnlnma ***\n" );
    printf( "\n     ** Input **\n\n" );

    fscanf( fp, "%d", &mx );
    fscanf( fp, "%d", &n );
    fscanf( fp, "%d", &m );

    printf( "\tmx=%3d\n", mx );
    printf( "\t n=%3d\n", n );
    printf( "\t m=%3d\n\n", m );

    x = ( double * )malloc((size_t)( sizeof(double) * (mx*(m+1)) ));
    if( x == NULL )
    {
        printf( "no enough memory for array x\n" );
        return -1;
    }

    b = ( double * )malloc((size_t)( sizeof(double) * ((m+1)*3) ));
    if( b == NULL )
    {
        printf( "no enough memory for array b\n" );
        return -1;
    }

    stat = ( double * )malloc((size_t)( sizeof(double) * ((m+1)*5) ));
    if( stat == NULL )
    {
        printf( "no enough memory for array stat\n" );
        return -1;
    }

    w1 = ( double * )malloc((size_t)( sizeof(double) * ((m+1)*(m+3)) ));
    if( w1 == NULL )
    {
        printf( "no enough memory for array w1\n" );
        return -1;
```

```
    }

    iw1 = ( int * )malloc((size_t)( sizeof(int) * (m+1) ));
    if( iw1 == NULL )
    {
        printf( "no enough memory for array iw1\n" );
        return -1;
    }

    for( j=0 ; j<m+1 ; j++ )
    {
        for( i=0 ; i<n ; i++ )
        {
            fscanf( fp, "%lf", &x[i+mx*j] );
        }
    }

    printf( "\t    y |          x\n");
    for( i=0 ; i<n ; i++ )
    {
        printf( "\t%5.3g | ",x[i+mx*m]  );
        for( j=0 ; j<m ; j++ )
        {
            printf( "%5.3g", x[i+mx*j] );
        }
        printf( "\n" );
    }

    fclose( fp );

    ierr = ASL_dnlnma(x, mx, m, n, b, r, v, stat, iw1, w1);

    printf( "\n     ** Output **\n\n" );
    printf( "\tierr = %6d\n\n", ierr );

    printf( "\tRegression coefficient\n" );

    printf( "\t    R.C.    S.E.    T-V.\n" );
    for( i=0 ; i<m+1 ; i++ )
    {
        printf( "\t" );
        for( j=0 ; j<3 ; j++ )
        {
            printf( "%8.3g", b[i+(m+1)*j] );
        }
        printf( "\n" );
    }
    printf( "\n" );

    printf( "\t         Multiple correlation coefficient=%8.3g\n", r[0] );
    printf( "\t                         Contribution ratio=%8.3g\n", r[1] );
    printf( "\tAdjusted multiple correlation coefficient=%8.3g\n\n", r[2] );

    printf( "\tAnalysis of variance table\n" );

    printf( "\t                S.S     D.F       U.V      F-R\n" );
    printf( "\tTotal      %8.3g  %6.1g\n", v[0],v[3] );
    printf( "\tRegression %8.3g  %6.1g  %8.3g %8.3g\n",
        v[1],v[4],v[6],v[8] );
    printf( "\tDeviation  %8.3g  %6.1g  %8.3g\n\n",
        v[2],v[5],v[7] );

    printf( "\tStatistics\n" );
    printf( "\t      Sum    Mean    S.S  Variance   S.D\n" );
    for( i=0 ; i<m+1 ; i++ )
    {
        printf( "\t" );
        for( j=0 ; j<5 ; j++ )
        {
            printf( "%8.3g", stat[i+(m+1)*j] );
        }
        printf( "\n" );
    }

    free( x );
    free( b );
    free( stat );
    free( iw1 );
    free( w1 );

    return 0;
}
```

(d) Output results

```
*** ASL_dnlnma ***

 ** Input **
```

```
mx=  5
 n=  5
 m=  3

    y |          x
   -1 |     2    1    3
    3 |     3   -1    0
    4 |    -2    0    2
    1 |     1   -2    2
   -1 |     3   -1    3

 ** Output **

ierr =      0

Regression coefficient
    R.C.    S.E.    T-V.
    5.21   0.898     5.8
  -0.744   0.192   -3.88
  0.0386   0.371   0.104
   -1.47   0.341   -4.31

         Multiple correlation coefficient=   0.985
                      Contribution ratio=   0.971
Adjusted multiple correlation coefficient=    0.94

Analysis of variance table
               S.S     D.F       U.V      F-R
Total         20.8       4
Regression    20.2       3      6.73     11.1
Deviation    0.608       1     0.608

Statistics
      Sum    Mean     S.S  Variance   S.D
        7     1.4    17.2       4.3  2.07
       -3    -0.6     5.2       1.3  1.14
       10       2       6       1.5  1.22
        6     1.2    20.8       5.2  2.28
```

10.3 NONLINEAR REGRESSION ANALYSIS

10.3.1 ASL_dnnlpo
Polynomial Regression Analysis

(1) **Function**

Assume that n independent variable values x_k $(k = 1, 2, \cdots, n)$ and n given dependent variable values y_k $(k = 1, 2, \cdots, n)$ corresponding to them have been given and obey the following linear regression model.

$$y_k = \beta_0 + \beta_1 x_k + \beta_2 {x_k}^2 + \cdots + \beta_m {x_k}^m + \varepsilon_k \quad (k = 1, 2, \cdots, n)$$

Here, ε_k are error terms that independently obey $N(0, \sigma^2)$.

The ASL_dnnlpo obtains:

(a) the estimates b_i $(i = 0, 1, \cdots, m)$ of partial correlation coefficients β_i $(i = 0, 1, \cdots, m)$

(b) the statistics for creating the following analysis of variance table due to the regression.

	Sum of squares	Degrees of freedom	Unbiased variance	F ratio
Total variation	S_T	$n-1$		
Variation due to regression	S_R	m	$V_R = \dfrac{S_R}{m}$	$F = \dfrac{V_R}{V_E}$
Residual variation	$S_E = S_T - S_R$	$n-m-1$	$V_E = \dfrac{S_E}{n-m-1}$	

(c) the multiple correlation coefficient R

$$R = \sqrt{\frac{S_R}{S_T}}$$

(d) the degree of freedom adjusted correlation coefficient R^*

$$R^* = \sqrt{1 - \frac{(n-1)S_E}{(n-m-1)S_T}}$$

(e) the standard deviation estimates ε_i $(i = 0, 1, \cdots, m)$ of the statistics b_i $(i = 0, 1, \cdots, m)$

(f) the test quantities $t_i (i = 0, 1, \cdots, m)$, which obey t distributions with $n - m - 1$ degrees of freedom based on the null hypotheses $H_0 : \beta_i = 0$ $(i = 0, 1, \cdots, m)$.

(2) **Usage**

Double precision:

ierr = ASL_dnnlpo (x, n, y, m, b, r, v, statx, staty, iw1, w1);

Single precision:

Nothing

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	x	D*	n	Input	Independent variate observed values x_k
2	n	I	1	Input	Number of observed values n
3	y	D*	n	Input	Dependent variate observed values y_i
4	m	I	1	Input	Polynomial degree m
5	b	D*	3×(m + 1)	Output	b[i]: β_i estimate b_i b[i+m+1]: β_i standard deviation estimate ε_i b[i+2×(m+1)]: β_i test quantity t value t_i ($i = 0, 1, \cdots,$ m)
6	r	D*	3	Output	r[0]: Multiple correlation coefficient R r[1]: Contribution ratio R^2 r[2]: Degree of freedom adjusted correlation coefficient R^*
7	v	D*	9	Output	Statistics for creating analysis of variance table (See Note (a))
8	statx	D*	5	Output	statx[0]: Independent variate sum statx[1]: Independent variate mean statx[2]: Independent variate sum of squares of deviations statx[3]: Independent variate variance statx[4]: Independent variate standard deviation
9	staty	D*	5	Output	staty[0]: Dependent variate sum staty[1]: Dependent variate mean staty[2]: Dependent variate sum of squares of deviations staty[3]: Dependent variate variance staty[4]: Dependent variate standard deviation
10	iw1	I*	m+1	Work	Work area
11	w1	D*	See Contents	Work	Work area **Size:** (m+1)×(m+4)
12	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) $2 \leq$ m + 1 < n

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
1000	Total sum of squares $\leq$ Regression sum of squares.	0.0 is set for b_i and t_i.
2000	Residual unbiased variance $\geq$ Total unbiased variance.	0.0 is set for R^*.
3000	Restriction (a) was not satisfied.	Processing is aborted.
4000	The inverse matrix could not be obtained.	

(6) **Notes**

(a) The following kind of a one-dimensional vector is stored in array v.

1:	S_T	Total variation
2:	S_R	Variation due to regression
3:	S_E	Residual variation
4:	$f_T = n - 1$	Degrees of freedom of total variation
5:	$f_R = m$	Degrees of freedom of variation due to regression
6:	$f_E = n - m - 1$	Degrees of freedom of residual variation
7:	V_R	Unbiased variance of variation due to regression
8:	V_E	Unbiased variance of residual variation
9:	F	F ratio

(7) **Example**

(a) Problem

Obtain the regression coefficient values, multiple correlation coefficient value, contribution ratio, analysis of variance table, and basic statistics according to a regression analysis from the following observation data.

x[0] =	1.0	y[0] =	10.0
x[1] =	3.0	y[1] =	20.0
x[2] =	5.0	y[2] =	25.0
x[3] =	6.0	y[3] =	26.0
x[4] =	8.0	y[4] =	36.0
x[5] =	10.0	y[5] =	62.0
x[6] =	11.0	y[6] =	78.0
x[7] =	13.0	y[7] =	107.0
x[8] =	14.0	y[8] =	118.0
x[9] =	15.0	y[9] =	127.0

(b) Input data

Observation data x and y, n=10 and m=4.

(c) Main program

```
/*      C interface example for ASL_dnnlpo */

#include <stdio.h>
#include <stdlib.h>
#include <asl.h>

int main()
{
    double *x, *y;
    int n, m;
    double *b, r[3], v[9];
    double statx[5], staty[5];
    int *iw1;
    double *w1;
    int ierr;

    int i,j;

    FILE *fp;


    fp = fopen( "dnnlpo.dat", "r" );
    if( fp == NULL )
    {
        printf( "file open error\n" );
        return -1;
    }

    printf( "    *** ASL_dnnlpo ***\n" );
    printf( "\n     ** Input **\n\n" );

    n=10;
    m=4;

    printf( "\tn=%3d\n", n );
    printf( "\tm=%3d\n\n", m );

    x = ( double * )malloc((size_t)( sizeof(double) * (n) ));
    if( x == NULL )
    {
        printf( "no enough memory for array x\n" );
        return -1;
    }

    y = ( double * )malloc((size_t)( sizeof(double) * (n) ));
    if( y == NULL )
    {
        printf( "no enough memory for array y\n" );
        return -1;
    }

    b = ( double * )malloc((size_t)( sizeof(double) * ((m+1)*3) ));
    if( b == NULL )
    {
        printf( "no enough memory for array b\n" );
        return -1;
    }

    iw1 = ( int * )malloc((size_t)( sizeof(int) * (n+1) ));
    if( iw1 == NULL )
    {
        printf( "no enough memory for array iw1\n" );
        return -1;
    }

    w1 = ( double * )malloc((size_t)( sizeof(double) * ((n+1)*(n+4)) ));
    if( w1 == NULL )
    {
        printf( "no enough memory for array w1\n" );
        return -1;
    }

    for( i=0 ; i<n ; i++ )
    {
        fscanf( fp, "%lf", &x[i] );
    }

    for( i=0 ; i<n ; i++ )
    {
        fscanf( fp, "%lf", &y[i] );
    }

    printf( "\tData of x and y\n");
    for( i=0 ; i<n ; i++ )
    {
        printf( "\t%8.3g  %8.3g\n", x[i] , y[i] );
    }

    fclose( fp );
```

```
    ierr = ASL_dnnlpo(x, n, y, m, b, r, v, statx, staty, iw1, w1);

    printf( "\n     ** Output **\n\n" );
    printf( "\tierr = %6d\n\n", ierr );

    printf( "\tRegression coefficient\n" );
    printf( "\t      R.C.      S.E.      T-V.\n" );
    for( i=0 ; i<m+1 ; i++ )
    {
        printf( "\t" );
        for( j=0 ; j<3 ; j++ )
        {
            printf( "  %8.3g", b[i+(m+1)*j] );
        }
        printf( "\n" );
    }
    printf( "\n" );

    printf( "\t         Multiple correlation coefficient=%8.3g\n", r[0] );
    printf( "\t                        Contribution ratio=%8.3g\n", r[1] );
    printf( "\tAdjusted multiple correlation coefficient=%8.3g\n\n", r[2] );

    printf( "\tAnalysis of variance table\n" );
    printf( "\t                    S.S     D.F          U.V      F-R\n" );
    printf( "\tTotal      %11.3e  %6.0f\n", v[0],v[3] );
    printf( "\tRegression %11.3e  %6.0f  %11.3e %8.3g\n",
        v[1],v[4],v[6],v[8] );
    printf( "\tDeviation  %11.3e  %6.0f  %11.3e\n\n",
        v[2],v[5],v[7] );

    printf( "\tStatistics\n" );
    printf( "\t          Sum     Mean      S.S   Variance    S.D\n" );
    printf( "\tx : " );
    for( i=0 ; i<5 ; i++ )
    {
        printf( "%9.3g", statx[i] );
    }
    printf( "\n" );
    printf( "\ty : " );
    for( i=0 ; i<5 ; i++ )
    {
        printf( "%9.3g", staty[i] );
    }
    printf( "\n" );

    free( x );
    free( y );
    free( b );
    free( iw1 );
    free( w1 );

    return 0;
}
```

(d) Output results

```
*** ASL_dnnlpo ***

 ** Input **

n= 10
m=  4

Data of x and y
       1        10
       3        20
       5        25
       6        26
       8        36
      10        62
      11        78
      13       107
      14       118
      15       127

 ** Output **

ierr =      0

Regression coefficient
      R.C.      S.E.      T-V.
     -4.61      1.57     -2.94
      18.9      3.03      6.24
     -4.97     0.764     -6.51
     0.551    0.0716      7.69
   -0.0176   0.00222     -7.92

         Multiple correlation coefficient=       1
                        Contribution ratio=   0.999
Adjusted multiple correlation coefficient=   0.999
```

```
Analysis of variance table
                  S.S     D.F          U.V      F-R
Total        1.744e+04       9
Regression   1.743e+04       4    4.357e+03 1.77e+03
Deviation    1.230e+01       5    2.461e+00

Statistics
          Sum     Mean      S.S   Variance    S.D
x :        86      8.6      206      22.9     4.79
y :       609     60.9 1.74e+04 1.94e+03       44
```

10.3.2 ASL_dnnlgf, ASL_rnnlgf
Regression According to an Arbitrary Function

(1) **Function**

Assume that n sets of independent variables $\boldsymbol{x}_k = (x_{1k}, x_{2k}, \cdots, x_{mk})$ $(k = 1, 2, \cdots, n)$ consisting of m variables and n given dependent variable values y_k $(k = 1, 2, \cdots, n)$ corresponding to them have been given and obey the following regression model.

$$y_k = f(\boldsymbol{x}_k; \boldsymbol{\beta}) + \varepsilon_k \ (k = 1, 2, \cdots, n)$$

Here, ε_k are error terms that independently obey $N(0, \sigma^2)$).

The ASL_dnnlgf or ASL_rnnlgf obtains:

(a) the estimates $\boldsymbol{b} = (b_1, b_2, \cdots, b_\ell)$ of the ℓ regression model parameters $\boldsymbol{\beta} = (\beta_1, \beta_2, \cdots, \beta_\ell)$

(b) the statistics for performing an analysis of variance for the regression model

	Sum of squares	Degrees of freedom	Unbiased variance
Total variation	S_T	n	
Residual variation	S_E	$n - \ell$	$V_E = \dfrac{S_E}{n - \ell}$

(c) the asymptotic variance-covariance matrix $V = (V_{ij})$ $(i, j = 1, \cdots, \ell)$

(d) the standard deviation estimates ε_i $(i = 1, \cdots, \ell)$ of the statistics b_i $(i = 1, \cdots, \ell)$

(2) **Usage**

Double precision:

```
ierr = ASL_dnnlgf  (f, xd, na, nn, nm, yd, nl, er, &nev, x, xe, y, c, nv, v, stat, iw1, w1);
```

Single precision:

```
ierr = ASL_rnnlgf  (f, xd, na, nn, nm, yd, nl, er, &nev, x, xe, y, c, nv, v, stat, iw1, w1);
```

(3) **Arguments and Return Value**

D:Double precision real Z:Double precision complex I: { int as for 32bit Integer / long as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
1	f	{D / R}	−	Input	Function name of the function f(x, b) for defining the regression model $f(\boldsymbol{x}, \boldsymbol{\beta})$.
2	xd	{D∗ / R∗}	na×nm	Input	Independent variate observation values $\boldsymbol{x}_k$ (See Note (a))
3	na	I	1	Input	Adjustable dimension of array x
4	nn	I	1	Input	Number of observed values m
5	nm	I	1	Input	Number of independent variables n
6	yd	{D∗ / R∗}	nn	Input	Dependent variate observation values y_i
7	nl	I	1	Input	Number of parameters ℓ of regression model function $f(\boldsymbol{x}; \boldsymbol{\beta})$.
8	er	{D / R}	1	Input	Requested precision (default value:$2\times\sqrt{\text{(units for error decision)}}$)
9	nev	I*	1	Input	Maximum number n of evaluations of function $f(\boldsymbol{x}, \boldsymbol{\beta})$ (default value:100×nn×nl)
				Output	Actual number of function evaluations
10	x	{D∗ / R∗}	nl	Input	Initial values of β_i
				Output	Estimated values b_i of β_i $(i =, 1, \cdots, \text{nl})$
11	xe	{D∗ / R∗}	nl	Output	Estimated standard deviation ε_i of b_i $(i = 1, \cdots, \text{nl})$
12	y	{D∗ / R∗}	nn	Output	Function value $f(\boldsymbol{x}_i, \boldsymbol{b})$ according to the optimal least square solution.

No.	Argument and Return Value	Type	Size	Input/ Output	Contents
13	c	$\left\{\begin{matrix}\text{D}*\\ \text{R}*\end{matrix}\right\}$	nv×nl	Output	Asymptotic variance-covariance matrix V for parameter $\boldsymbol{\beta}$.
14	nv	I	1	Input	Adjustable dimension of array cv.
15	v	$\left\{\begin{matrix}\text{D}*\\ \text{R}*\end{matrix}\right\}$	5	Output	Statistics for creating analysis of variance table. (See Note (c))
16	stat	$\left\{\begin{matrix}\text{D}*\\ \text{R}*\end{matrix}\right\}$	(nl+1)×5	Output	Basic statistics of calculation result. (See Note (b))
17	iw1	I*	4 × nl	Work	Work area
18	w1	$\left\{\begin{matrix}\text{D}\\ \text{R}\end{matrix}\right\}$	See Contents	Work	Work area **Size:**nn × (2 × nl + 1) + nl × (nl + 4)
19	ierr	I	1	Output	Error indicator (Return Value)

(4) **Restrictions**

(a) $0 < \text{nn} \leq \text{na}$

(b) $0 < \text{nl} \leq \text{nv}$

(c) $\text{nm} > 0$

(d) $2 \leq \text{nl} + 1 < \text{nn}$

(5) **Error indicator (Return Value)**

ierr value	Meaning	Processing
0	Normal termination.	
3000	Any of restrictions (a) to (d) was not satisfied.	Processing is aborted.
4000	The linear least squares method could not be solved.	The values of x, y, stat and v at that time are output.
4100	The steepest descent could not be calculated.	
4200	The solution could not be corrected 2nn times consecutively.	
4300	The inverse matrix could not be obtained.	
5000	The sequence did not converge before reaching the given maximum number of evaluations.	

(6) **Notes**

(a) Observation data x_{ki} $(k = 1, 2, \cdots, n; i = 1, 2, \cdots, m)$ are stored as a real matrix (two-dimensional array type) in array xd as follows (See Appendix A).

$$\begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1,m} \\ x_{21} & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ x_{n1} & \cdots & \cdots & x_{nm} \end{bmatrix}$$

(b) The following kind of real matrix (two-dimensional array type) is stored in array stat (See Appendix A).

$$\begin{bmatrix} \sum_{k=1}^{n} x_{k1} & \mu_1 & \sum_{k=1}^{n}(x_{k1}-\mu_1)^2 & \dfrac{\sum_{k=1}^{n}(x_{k1}-\mu_1)^2}{n-1} & \sqrt{\dfrac{\sum_{k=1}^{n}(x_{k1}-\mu_1)^2}{n-1}} \\ \sum_{k=1}^{n} x_{k2} & \mu_2 & \sum_{k=1}^{n}(x_{k2}-\nu)^2 & \dfrac{\sum_{k=1}^{n}(x_{k2}-\mu_2)^2}{n-1} & \sqrt{\dfrac{\sum_{k=1}^{n}(x_{k2}-\mu_2)^2}{n-1}} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \sum_{k=1}^{n} x_{km} & \mu_1 & \sum_{k=1}^{n}(x_{km}-\nu)^2 & \dfrac{\sum_{k=1}^{n}(x_{km}-\mu_m)^2}{n-1} & \sqrt{\dfrac{\sum_{k=1}^{n}(x_{km}-\mu_m)^2}{n-1}} \\ \sum_{k=1}^{n} y_k & \nu & \sum_{k=1}^{n}(y_k-\nu)^2 & \dfrac{\sum_{k=1}^{n}(y_k-\nu)^2}{n-1} & \sqrt{\dfrac{\sum_{k=1}^{n}(y_k-\nu)^2}{n-1}} \end{bmatrix}$$

Here, μ_i and ν are defined as follows.

$$\mu_i = \frac{\sum_{k=1}^{n} x_{ki}}{n}$$

$$\nu = \frac{\sum_{k=1}^{n} y_k}{n}$$

(c) The following kind of a one-dimensional vector is stored in array v.

1:	S_T	Total variation
2:	S_E	Residual variation
3:	$f_T = n$	Degrees of freedom of total variation
4:	$f_E = n - \ell$	Degrees of freedom of residual variation
5:	V_E	Unbiased variance of residual variation

(d) The function f should be created as follows.

```
double FORTRAN f(double *x, double *b)
{
    return f(x, b);
}
```

(e) The convergence decision is made according to the following expression by solving for $\boldsymbol{b} + \Delta\boldsymbol{b}$.

$$\|\Delta\boldsymbol{b}\| \leq \mathrm{er} \times \max(1, \|\boldsymbol{b} + \Delta\boldsymbol{b}\|)$$

Here, $\Delta\boldsymbol{b}$ is the correction vector for $\boldsymbol{b}$ and $\|\boldsymbol{b}\| = \max|b_i|$
A value on the order of the default value should be taken as er.

(f) If a default value is written in the Contents column of the Arguments table, the default value is set if an integer less than or equal to 0 is entered for an integer type or a real number less than or equal to 0.0 is entered for a real type.

(7) **Example**

(a) Problem
From the following observation data,

$$(x_{ki}|y_k) = \left[\begin{array}{rrr|l} -2.0 & -2.0 & -1.0 & 2.7 \\ -1.5 & -1.5 & -1.0 & 2.9 \\ -1.0 & -1.0 & -1.0 & 3.1 \\ -0.5 & -1.0 & -1.5 & 3.4 \\ -0.5 & -1.5 & 1.0 & 3.9 \\ 0.0 & 0.0 & 0.0 & 4.7 \\ 0.5 & 0.25 & 0.25 & 6.0 \\ 0.5 & 1.0 & 0.5 & 7.8 \\ 1.0 & 1.0 & 1.0 & 7.9 \\ 1.5 & 1.5 & 1.0 & 6.3 \\ 1.5 & 2.0 & 1.5 & 5.2 \end{array}\right]$$

perform a regression analysis according to the following regression model

$$f(\boldsymbol{x};\boldsymbol{\beta}) = \frac{\beta_3\beta_2^2}{(x_1+x_2+x_3-\beta_1)^2+\beta_2^2} + \beta_4 + \beta_5(x_1+x_2+x_3)$$

to obtain the estimates and estimate errors of the regression parameters, asymptotic variance-covariance matrix, analysis of variance table, and basic statistics.

(b) Input data
Observation data xd and yd, nn=11, nm=3, nl=5, nev=0, initial values of regression parameters, nev=0 and er=0.0.

(c) Main program

```
/*      C interface example for ASL_dnnlgf */

#include <stdio.h>
#include <stdlib.h>
#include <asl.h>

#ifdef __cplusplus
extern "C" {
#endif
double f(double *x,double *a);
#ifdef __cplusplus
}
#endif

double f(double *x,double *a)
{
    double f1,f2;
    f1=a[2]*a[1]*a[1]/(((x[0]+x[1]+x[2])-a[0])
       *((x[0]+x[1]+x[2])-a[0])+(a[1]*a[1]));
    f2=a[3]+a[4]*(x[0]+x[1]+x[2]);
    return (f1+f2);
}

int main()
{
```

```
double *xd;
double *yd;
int n;
double er;
int nev;
double *x;
double *xe;
int m;
int l;
int na;
int nv;
double *y;
double *c;
double *v;
int *iwk;
double *stat;
double *wk;
int ierr;
int i,j;
FILE *fp;

fp = fopen( "dnnlgf.dat", "r" );
if( fp == NULL )
{
    printf( "file open error\n" );
    return -1;
}

printf( "    *** ASL_dnnlgf ***\n" );
printf( "\n     ** Input **\n\n" );
n=11;
l=5;
m=3;
na=11;
nv=5;
xd  = ( double * )malloc((size_t)( sizeof(double) * (na*m) ));
if( xd  == NULL )
{
    printf( "no enough memory for array xd\n" );
    return -1;
}
yd  = ( double * )malloc((size_t)( sizeof(double) * n ));
if( yd  == NULL )
{
    printf( "no enough memory for array yd\n" );
    return -1;
}
x  = ( double * )malloc((size_t)( sizeof(double) * l ));
if( x  == NULL )
{
    printf( "no enough memory for array x\n" );
    return -1;
}
xe  = ( double * )malloc((size_t)( sizeof(double) * l ));
if( xe  == NULL )
{
    printf( "no enough memory for array xe\n" );
    return -1;
}
y  = ( double * )malloc((size_t)( sizeof(double) * n ));
if( y  == NULL )
{
    printf( "no enough memory for array y\n" );
    return -1;
}
c  = ( double * )malloc((size_t)( sizeof(double) * (nv*l) ));
if( c  == NULL )
{
    printf( "no enough memory for array c\n" );
    return -1;
}
v  = ( double * )malloc((size_t)( sizeof(double) * 5 ));
if( v  == NULL )
{
    printf( "no enough memory for array v\n" );
    return -1;
}

stat  = ( double * )malloc((size_t)( sizeof(double) * 5 * (m+1) ));
if( stat  == NULL )
{
    printf( "no enough memory for array stat\n" );
    return -1;
}

iwk  = ( int * )malloc((size_t)( sizeof(int) * (4*l) ));
if( iwk  == NULL )
{
    printf( "no enough memory for array iwk\n" );
    return -1;
}

wk  = ( double * )malloc((size_t)( sizeof(double) * (n*(2*l+1)+l*(l+4)) ));
```

```
if( wk  == NULL )
{
    printf( "no enough memory for array wk\n" );
    return -1;
}

for( i=0 ; i<n ; i++ )
{
    for( j=0 ; j<m ; j++ ){
        fscanf( fp, "%lf", &xd[i+j*na] );
    }
    fscanf( fp, "%lf", &yd[i] );
}
fscanf( fp, "%d", &nev );
fscanf( fp, "%lf", &er );
for( i=0 ; i<l ; i++ )
{
    fscanf( fp, "%lf", &x[i] );
}
printf( "\tn  =  %6d\n", n );
printf( "\tm  =  %6d\n", m );
printf( "\tl  =  %6d\n", l );
printf( "\tnev=  %6d\n", nev );
printf( "\ter =%8.3g\n", er );
printf( "\tnv =  %6d\n", nv );
printf( "\n\tCoordinates (x,y)\n\n" );
printf( "\t\t          x          \t      y\n\n" );
for( i=0 ; i<n ; i++ )
{
    printf(  "\t");
    for( j=0 ; j<m ; j++ ){
        printf(  "%8.3g ",xd[i+j*na] );
    }
    printf(  "\t%8.3g\n",yd[i] );
}
printf( "\n\tInitial Value of Coefficients\n\n" );
for( i=0 ; i<l ; i++ )
{
    printf(  "\t  a[%6d]=%8.3g\n", i,x[i] );
}

fclose( fp );

ierr = ASL_dnnlgf(f,xd,na,n,m,yd,l,er,&nev,x,xe,y,c,nv,v,stat,iwk,wk);

printf( "\n     ** Output **\n\n" );
printf( "\tierr = %6d\n", ierr );
printf( "\n\tnev  = %6d\n", nev  );
printf( "\n\tOptimized Coefficients\n\n" );
for( i=0 ; i<l ; i++ )
{
    printf(  "\t  a[%6d]=%8.3g\n", i,x[i] );
}
printf( "\n\tEstimated Errors of Coefficients\n\n" );
for( i=0 ; i<l ; i++ )
{
    printf(  "\t  ae[%6d]=%8.3g\n", i,xe[i] );
}
printf( "\n\tFunction Value\n\n" );
for( i=0 ; i<n ; i++ )
{
    printf(  "\t  yf[%6d]=%8.3g\n", i,y[i] );
}
printf( "\n\tAsymptotic Variance-Covariance matrix\n\n" );
for( i=0 ; i<l ; i++ )
{
    printf(  "\t" );
    for( j=0 ; j<l ; j++ )
    {
        printf( "%8.3g ", c[i+j*nv] );
    }
    printf(  "\n" );
}
printf( "\n\tAnalysis of variance table\n\n" );
for( i=0 ; i<5 ; i++ )
{
    printf(  "\t  v[%6d]=%8.3g\n", i,v[i] );
}
printf( "\n\tStatistics\n\n" );
for( i=0 ; i<m ; i++ )
{
    printf(  "\t  x[%6d]:\t", i );
    for( j=0 ; j<5 ; j++ )
    {
        printf( "%8.3g ", stat[i+j*(m+1)] );
    }
```

```
        printf(  "\n" );
    }
    printf(  "\t  y[%6d]:\t", i );
    for( j=0 ; j<5 ; j++ )
    {
        printf( "%8.3g ", stat[i+j*(m+1)] );
    }
    printf(  "\n", i );

    free( xd );
    free( yd );
    free( x );
    free( xe );
    free( y );
    free( c );
    free( v );
    free( stat );
    free( iwk );
    free( wk );

    return 0;
}
```

(d) Output results

```
*** ASL_dnnlgf ***

 ** Input **

n  =      11
m  =       3
l  =       5
nev=       0
er =       0
nv =       5

Coordinates (x,y)

               x                    y

      -2       -2       -1       2.7
    -1.5     -1.5       -1       2.9
      -1       -1       -1       3.1
    -0.5       -1     -1.5       3.4
    -0.5     -1.5        1       3.9
       0        0        0       4.7
     0.5     0.25     0.25         6
     0.5        1      0.5       7.8
       1        1        1       7.9
     1.5      1.5        1       6.3
     1.5        2      1.5       5.2

Initial Value of Coefficients

  a[     0]=       0
  a[     1]=       1
  a[     2]=       6
  a[     3]=     3.5
  a[     4]=     0.2

 ** Output **

ierr =      0

nev  =    902

Optimized Coefficients

  a[     0]=    2.49
  a[     1]=    1.75
  a[     2]=    4.86
  a[     3]=    3.07
  a[     4]=   0.113

Estimated Errors of Coefficients

  ae[     0]=   0.105
  ae[     1]=   0.317
  ae[     2]=   0.523
  ae[     3]=   0.413
  ae[     4]=  0.0695

Function Value

  yf[     0]=    2.76
  yf[     1]=    2.95
  yf[     2]=    3.18
  yf[     3]=    3.18
  yf[     4]=    3.93
  yf[     5]=    4.68
  yf[     6]=       6
  yf[     7]=    7.81
```

```
  yf[     8]=    7.89
  yf[     9]=     6.3
  yf[    10]=    5.22

Asymptotic Variance-Covariance matrix

  0.0111   0.0183   0.0315  -0.0254 -0.00491
  0.0183      0.1    0.115    -0.12  -0.0182
  0.0315    0.115    0.274   -0.191  -0.0321
 -0.0254    -0.12   -0.191     0.17   0.0261
-0.00491  -0.0182  -0.0321   0.0261  0.00483

Analysis of variance table

  v[      0]=      300
  v[      1]=   0.0628
  v[      2]=       11
  v[      3]=        8
  v[      4]= 0.00785

Statistics

  x[      0]:        -0.5  -0.0455     13.7     1.37     1.17
  x[      1]:       -1.25   -0.114     18.7     1.87     1.37
  x[      2]:        0.75   0.0682     10.8     1.08     1.04
  y[      3]:        53.9      4.9       36      3.6      1.9
```

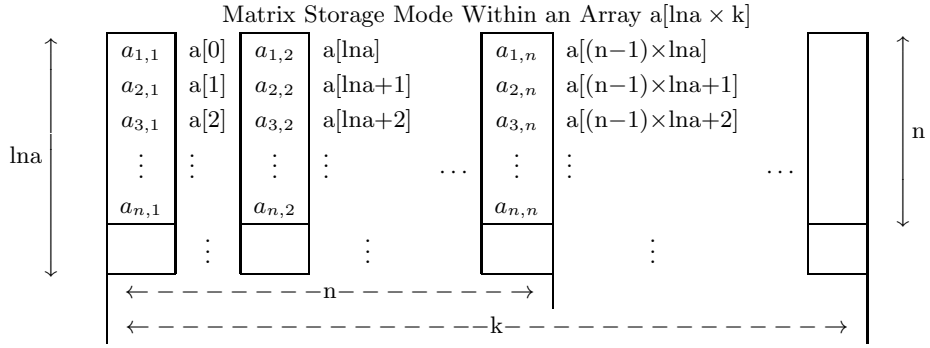
Appendix A

METHODS OF HANDLING ARRAY DATA

A.1 Methods of handling array data corresponding to matrix

Since the ASL C interface function library uses array data corresponding to matrix, this section describes various methods of handling arrays.

To call a function that uses array data, you must declare that array in advance in the calling program. If the declared array is a[lna × k], then $n \times n$ matrix $A = (a_{i,j})$ $(i = 1, 2, \cdots, n; j = 1, 2, \cdots, n)$ is stored in array a as shown in the figure below.



Remarks

a. lna $\geq n$ and k $\geq n$ must hold.

b. Matrix element $a_{i,j}$ corresponds to the array element a$[(i-1) + \text{lna} \times (j-1)]$.

Figure A−1 Matrix Storage Mode Within an Array a[lna × k]

lna is called an adjustable dimension. If a two-dimensional array is used as an argument, the adjustable must be passed to the function as an argument in addition to the array name and order of the array. Because the matrix storage mode in ASL C interface corresponds to that of FORTRAN and the matrix elements $a_{i,j}$ $(i = 1, 2, \cdots, \text{lna}; j = 1, 2, \cdots, \text{k})$ must correspond to the array element a$[\ell]$ $(\ell = 0, 1, 2, \cdots, \text{lna} \times \text{k} - 1)$, as follows on the main memory.

$a_{1,1}$	$a_{2,1}$	$\cdots$	$a_{\text{lna},1}$	$a_{1,2}$	$a_{2,2}$	$\cdots$
↕	↕	$\cdots$	↕	↕	↕	$\cdots$
a[0]	a[1]	$\cdots$	a[lna − 1]	a[lna]	a[lna + 1]	$\cdots$

Example ASL_dam1ad (Real matrix addition)

Add 3×2 matrices A and B placing the sum in matrix C. If you declare arrays of size $[5 \times 4]$, the declaration is as follows.

```
/*      C interface example for ASL_dam1ad */
#include <stdio.h>
#include <stdlib.h>

#include <asl.h>

int main()
{
    double *a, *b, *c;
    int lma, lmb, lmc;
    int m, n, ierr;
```

```
    int k;
    lma = lmb = lmc = 5;
    k = 4;
    m = 3;
    n = 2;
    a = (double *)malloc((size_t) sizeof(double) * lma*k);
    if(a == NULL)
    {
        printf("no enough memory for array a\n");
        return -1;
    }
    b = (double *)malloc((size_t) sizeof(double) * lmb*k);
    if(b == NULL)
    {
        printf("no enough memory for array b\n");
        return -1;
    }
    c = (double *)malloc((size_t) sizeof(double) * lmc*k);
    if(c == NULL)
    {
        printf("no enough memory for array c\n");
        return -1;
    }

          ⋮
    ierr =  ASL _dam1ad(a, lma , m, n, b, lmb, c,  lmc);
          ⋮

    free(a);
    free(b);
    free(c);
    return 0;
}
```

Data is stored in a as follows. Data are stored in b and c in the same way.

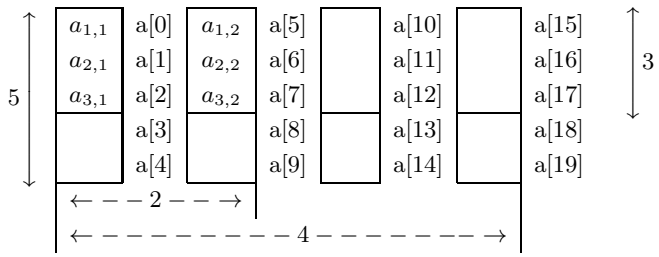


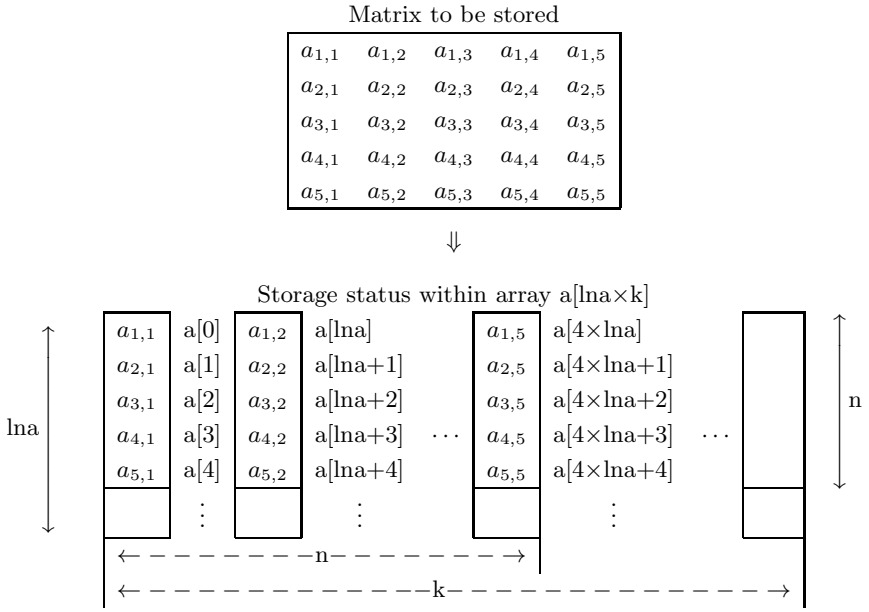
Figure A−2 Matrix Storage Mode Within an Array a[lna × k]

If you will be manipulating several arrays having different orders as data, you can prepare one array having lna equal to the largest order and use that array successively for each array. However, you must always assign the lna value as an adjustable dimension.

A.2 Data storage modes

Matrix data storage modes differ according to the matrix type. Storage modes for each type of matrix are shown below.

A.2.1 Real matrix (two-dimensional array type)



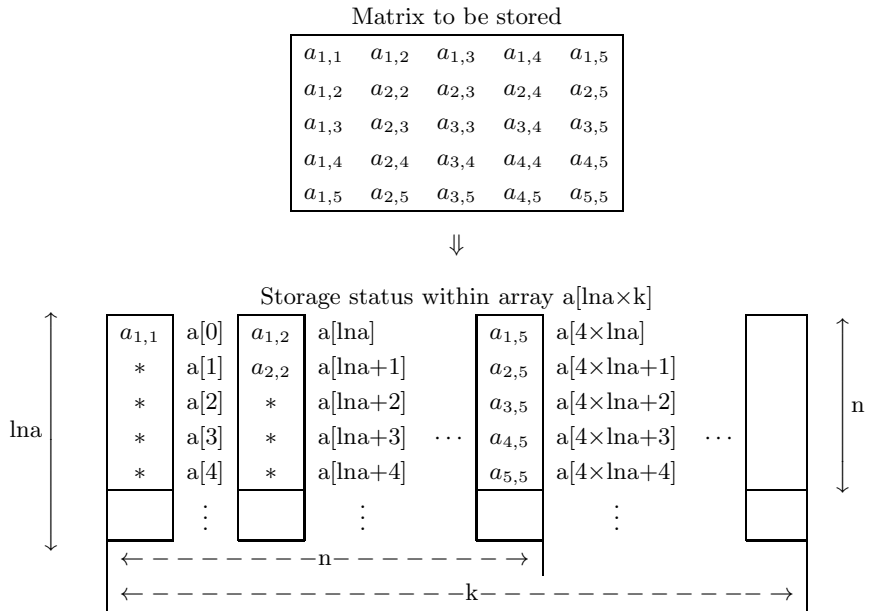
Remarks

a. lna ≥ n and k ≥ n must hold.

Figure A−3 Real Matrix (Two-Dimensional Array Type) Storage Mode

A.2.2 Real symmetric matrix and positive symmetric matrix

(1) **Two-dimensional array type, upper triangular type**



Remarks

a. The asterisk (∗) indicates an arbitrary value.

b. lna ≥ n and k ≥ n must hold.

Figure A−4 Real Symmetric Matrix (Two-dimensional Array Type) (Upper Triangular Type) Storage mode

Appendix B

MACHINE CONSTANTS USED IN ASL C INTERFACE

B.1 Units for Determining Error

The table below shows values in ASL C interface as units for determining error in floating point calculations. The units shown in the table are numeric values determined by the internal representation of floating point data. ASL C interface uses these units for determining convergence and zeros.

Table B−1 Units for Determining Error

Single-precision	Double-precision
$2^{-23}(\simeq 1.19 \times 10^{-7})$	$2^{-52}(\simeq 2.22 \times 10^{-16})$

Remark: The unit for determining error ε, which is also called the machine ε, is usually defined as the smallest positive constant for which the calculation result of $1+\varepsilon$ differs from 1 in the corresponding floating point mode. Therefore, seeing the unit for determining error enables you to know the maximum number of significant digits of an operation (on the mantissa) in that floating point mode.

B.2 Maximum and Minimum Values of Floating Point Data

The table below shows maximum and minimum values of floating point data defined within ASL C interface. Note that the maximum and minimum values shown below may differ from the maximum and minimum values that are actually used by the hardware for each floating point mode.

Table B−2 Maximum and Minimum Values of Floating Point Data

	Single-precision	Double-precision
Maximum value	$2^{127}(2-2^{-23})$ $(\simeq 3.40 \times 10^{38})$	$2^{1023}(2-2^{-52})$ $(\simeq 1.80 \times 10^{308})$
Positive minimum value	2^{-126} $(\simeq 1.17 \times 10^{-38})$	2^{-1022} $(\simeq 2.23 \times 10^{-308})$
Negative maximum value	-2^{-126} $(\simeq -1.17 \times 10^{-38})$	-2^{-1022} $(\simeq -2.23 \times 10^{-308})$
Minimum value	$-2^{127}(2-2^{-23})$ $(\simeq -3.40 \times 10^{38})$	$-2^{1023}(2-2^{-52})$ $(\simeq -1.80 \times 10^{308})$

Index

(*) SMP Functions = Shared Memory Parallel Processing Functions