ADVANCED SCIENTIFIC LIBRARY
ASL
User's Guide
<Basic Functions Vol.1>

PREFACE

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

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 1.

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 subroutine related to storage mode conversion of array data.
3	Basic Matrix Algebra	Explanation of algorithms, method of using, and usage example of subroutine related to basic calculations involving matrices.
4	Eigenvalues and Eigenvectors	Explanation of algorithms, method of using, and usage example of subroutine 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 subroutine 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 subroutine 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 subroutine 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 subroutine 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 subroutine 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 subroutine 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 subroutine 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 subroutine 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 subroutine related to sorting and ranking.
4	Roots of Equations	Explanation of algorithms, method of using, and usage example of subroutine 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 subroutine 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 subroutine related to uniform random number tests, and distribution random number tests.
3	Probability Distributions	Explanation and usage examples of subroutine related to continuous distributions and discrete distributions.
4	Basic Statistics	Explanation and usage examples of subroutine related to basic statistics, variance-covariance and correlation.
5	Tests and Estimates	Explanation and usage examples of subroutine related to interval estimates and tests.
6	Analysis of Variance and Design of Experiments	Explanation and usage examples of subroutine 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 subroutine related to tests using χ^2 distribution and tests using other distributions.
8	Multivariate Analysis	Explanation and usage examples of subroutine related to principal component analysis, factor analysis, canonical correlation analysis, discriminant analysis, cluster analysis.
9	Time Series Analysis	Explanation and usage examples of subroutine related to autocorrelation, cross correlation, autocovariance, cross covariance, smoothing and demand forecasting.
10	Regression analysis	Explanation and usage examples of subroutine 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 subroutine 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 subroutine 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 subroutine 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 subroutine 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 subroutine 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 subroutine 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

Table 1−1 shows correspondences among product categories, functions of ASL and supported hardware platforms. In the same version of ASL, interfaces of subroutines of the same name 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

ASL has the following distinctive characteristics.

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

(2) Special-purpose subroutines 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 subroutines.

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

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

1.2 KINDS OF LIBRARIES

Table 1−2 Kinds of libraries providing ASL

Size of variable(byte)		Declaration of arguments	Kind	Kind of library
integer	real			

4	8	INTEGER(4)
		REAL(8)
			32bit integer Double-precision subroutine	32bit integer library
				(link option: -lasl_sequential)

4	4	INTEGER(4)
		REAL(4)
			32bit integer Single-precision subroutine	

8	8	INTEGER(8)
		REAL(8)
			64bit integer Double-precision subroutine	64bit integer library
				(link option: -lasl_sequential_i64)

8	4	INTEGER(8)
		REAL(4)
			64bit integer Single-precision subroutine	

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

(∗2) The string "(4)" that specifies 32bit (4 byte) can be omitted.

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 such information as the effective ways of using the subroutines, techniques employed, algorithms on which the subroutines are based, and notes.

1.3.2 Organization of Subroutine Description

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

(1) Function

(2) Usage

(3) Arguments

(4) Restrictions

(5) Error indicator

(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 subroutine.

(2) **Usage**

Usage describes the subroutine name and the order of its arguments. In general, arguments are arranged as follows.

CALL subroutine-name (input-arguments, input/output-arguments, output-arguments, ISW, work, IERR)

ISW is an input argument for specifying the processing procedure. IERR is an error indicator. 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**

Arguments are explained in the order described above in paragraph (2). The explanation format is as follows.

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

(a) Arguments

Arguments 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 subroutine, 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 `INTEGER/ INTEGER(8)`, 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 subroutine.

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 $3 \times \mathrm{N}$ or $\mathrm{N} + \mathrm{M}$.

M**,** N : Indicates that the argument is a two-dimensional array having M rows and N columns. If M and N indicating the size of this array have not been defined before this array is specified, they are defined as arguments immediately following this array.

(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 subroutine, information when the argument is input is preserved. The user must assign input-time information unless specifically instructed otherwise.

ii. When only “Output” appears

Results calculated within the subroutine are output to the argument. No data is entered at input time.

iii. When both “Input” and “Output” appear

Argument contents change between the time control passes to the subroutine and the time control returns from the subroutine. The user must assign input-time information unless specifically instructed otherwise.

iv. When “Work” appears

Work indicates that the argument is an area used when performing calculations within the subroutine. A work area having the specified size must be reserved in the program calling this subroutine. 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 subroutine (DBGMLC, RBGMLC) that obtains the LU decomposition and the condition number of a real matrix is as follows.

Double precision:
CALL DBGMLC (A, LNA, N, IPVT, COND, W1, IERR)
Single precision:
CALL RBGMLC (A, LNA, N, IPVT, COND, W1, IERR)

The explanation of the arguments is as follows.

Table 1−3 Sample Arguments

D:Double precision real Z:Double precision complex I: $\left\{\begin{matrix}\text{INTEGER(4) as for 32bit Integer}\\\text{INTEGER(8) as for 64bit Integer}\end{matrix}\right\}$
R:Single precision real C:Single precision complex

No.	Argument	Type	Size	Input/ Output	Contents
1	A	Note $\left\{\begin{matrix}\text{D}\\\text{R}\end{matrix}\right\}$	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): Number of the row exchanged with row i in the i-th step.
5	COND	$\left\{\begin{matrix}\text{D}\\\text{R}\end{matrix}\right\}$	1	Output	Reciprocal of the condition number
6	W1	$\left\{\begin{matrix}\text{D}\\\text{R}\end{matrix}\right\}$	N	Work	Work area
7	IERR	I	1	Output	Error indicator

To use this subroutine, arrays A, IPVT and W1 must first be allocated in the calling program so they can be used as arguments. A is a $\left\{\begin{matrix}\text{double-precision}\\\text{single-precision}\end{matrix}\right\}^{\text{Note}}$ real array of size (LNA , N) , IPVT is an integer array of size N and W1 is a $\left\{\begin{matrix}\text{double-precision}\\\text{single-precision}\end{matrix}\right\}$ 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 `INTEGER(8)`, not `INTEGER`.

Note The entries enclosed in brace { } mean that the array should be declared double precision type (code D) when using subroutine DBGMLC and real type (code R) when using subroutine 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 subroutine is called. The LU decomposition and condition number of the assigned matrix are calculated with in the subroutine, and the results are stored in array A and variable COND. In addition, pivoting information is stored in IPVT for use by subsequent subroutines.

IERR is an argument 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 subroutine, its contents at input and output time have no special meaning.

(4) Restrictions

Restrictions indicate limiting ranges for subroutine arguments.

(5) Error indicator

Each subroutine has been given an error indicator as an output argument. 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 subroutine, a corresponding value is output to IERR. Error indicator values are divided into five levels.

Table 1−4 Classification of Error Indicator Output Values

Level	IERR 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 subroutine.

(7) Example

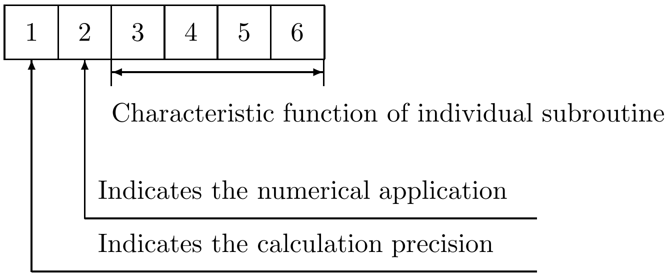
Here gives an example of how to use the subroutine. Note that in some cases, multiple subroutines are combined in a single example. The output results are given in the 32-bit integer version, and may differ within the range of rounding error if the compiler or intrinsic functions are different.

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 SUBROUTINE NAMES

The subroutines name of ASL basic functions consists of ⟨six alphanumeric characters⟩.

Figure 1−1 Subroutine Name Components


1
2
3
4
5
6
Characteristic function of individual subroutine
Indicates the numerical application
Indicates the calculation precision

"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 letterererere are used to indicate the application field in the ASL 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
S	Sorting and ranking	5, 7

Letter	Application Field	Volume
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 subroutine.

1.5 NOTES

(1) Use the subroutines 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.

(2) To suppress compiler operation exceptions, ASL subroutines 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.

(3) 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.

(4) 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.

(5) 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.

(6) 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 subroutine.

(7) 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.

(8) 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

STORAGE MODE CONVERSION

2.1 INTRODUCTION

This chapter describes subroutines that perform storage mode conversions of matrices.

Since this library uses various storage modes that differ according to the type and characteristics of the matrix, you must store the matrix in advance in the storage mode that corresponds to the subroutine to be used. If the matrix has already been stored, you must change its storage mode. Mode conversion subroutines have been provided to facilitate this process.

2.1.1 Algorithms Used

2.1.1.1 Real band matrix compression and restoration

The element in the i-th row j-th column of the real band matrix is stored as follows.

$$\begin{array}{lcl} \text{Matrix} & & \text{Band type} \\ A_{i,j} & \longleftrightarrow & \mathrm{A(j-i+ML+1,i)} \end{array}$$

Remarks

a. ML is the lower band width.

2.1.1.2 Real symmetric band matrix compression and restoration

The element in the i-th row j-th column of the real symmetric band matrix is stored as follows.

$$\begin{array}{lcl} \text{Matrix} & & \text{Symmetric band type} \\ A_{i,j} & \longleftrightarrow & \mathrm{A(i-j+MB+1,j)} \end{array}$$

Remarks

a. MB is the band width.

2.1.1.3 One-dimensional column-oriented list format storage of a sparse matrix

The element in the i-th row j-th column of the sparse matrix is stored as follows.

$$\begin{array}{lcl} \text{Matrix} & & \text{One–dimensional column-oriented list format} \\ A_{i,j} & \longrightarrow & \left\{ \begin{array}{rcl} k & = & \mathrm{IPONTR(j)+m} \\ i & = & \mathrm{IRWIND(k)} \\ A_{i,j} & = & \mathrm{VALUES(k)} \end{array} \right. \end{array}$$

Remarks

a. Asymmetric matrix storage:
ITYPE = 1.
The parameter m denotes an ordering number that is allocated to each nonzero element in the j -th column of a given matrix, which begins with the value 0.

b. Symmetric matrix storage, using the upper triangular part as input ITYPE = 2.
The parameter m denotes an ordering number that is allocated to each nonzero element in the j -th column of the upper triangular part of a given matrix, which begins with the value 0.

c. Symmetric matrix storage, using the lower triangular part as input ITYPE = 2.
The parameter m denotes an ordering number that is allocated to each nonzero element in the j -th column of the lower triangular part of a given matrix, which begins with the value 0.

2.1.1.4 ELLPACK format of sparse matrix

The element in the i-th row j-th column of the sparse matrix is stored as follows.

$$\begin{array}{lcl} \text{Matrix} & & \text{ELLPACK format} \\ A_{i,j} & \longrightarrow & \left\{ \begin{array}{rcl} A_{i,j} & = & \mathrm{A(i,m)} \\ j & = & \mathrm{JA(i,m)} \end{array} \right. \end{array}$$

Remarks

a. The parameter m denotes an ordering number that is allocated to each nonzero element in the i -th row of a given matrix, which begins with the value 1. The smallest value $m = 1$ should always be given to the diagonal element. As for other elements, values $m = 2, 3, \ldots$ can be allocated to them in an arbitrary order.

2.2 STORAGE MODE CONVERSION

2.2.1 DABMCS, RABMCS
Storage Mode Conversion of a Real Band Matrix: from (Two-Dimensional Array Type) to (Band Type)

(1) **Function**

DABMCS or RABMCS converts the storage mode of the real band matrix A from (two-dimensional array type) to (band type).

(2) **Usage**

Double precision:
  CALL DABMCS (A, LNA, N, MU, ML, B, LMB, IERR)
Single precision:
  CALL RABMCS (A, LNA, N, MU, ML, B, LMB, IERR)

(3) **Arguments**

D:Double precision real Z:Double precision complex I: $\left\{\begin{array}{l}\text{INTEGER(4) as for 32bit Integer}\\ \text{INTEGER(8) as for 64bit Integer}\end{array}\right\}$
R:Single precision real C:Single precision complex

No.	Argument	Type	Size	Input/ Output	Contents
1	A	$\left\{\begin{array}{c}\text{D}\\ \text{R}\end{array}\right\}$	LNA, N	Input	Real band matrix A (two-dimensional array type) (See Appendix B).
2	LNA	I	1	Input	Adjustable dimension of array A.
3	N	I	1	Input	Order of matrix A.
4	MU	I	1	Input	Upper band width of matrix A.
5	ML	I	1	Input	Lower band width of matrix A.
6	B	$\left\{\begin{array}{c}\text{D}\\ \text{R}\end{array}\right\}$	LMB, N	Output	Real band matrix A (band type) (See Appendix B).
7	LMB	I	1	Input	Adjustable dimension of array B.
8	IERR	I	1	Output	Error indicator

(4) **Restrictions**

(a) $0 \leq \text{MU} < \text{N}$

(b) $0 \leq \text{ML} < \text{N}$

(c) $0 < \text{N} \leq \text{LNA}$

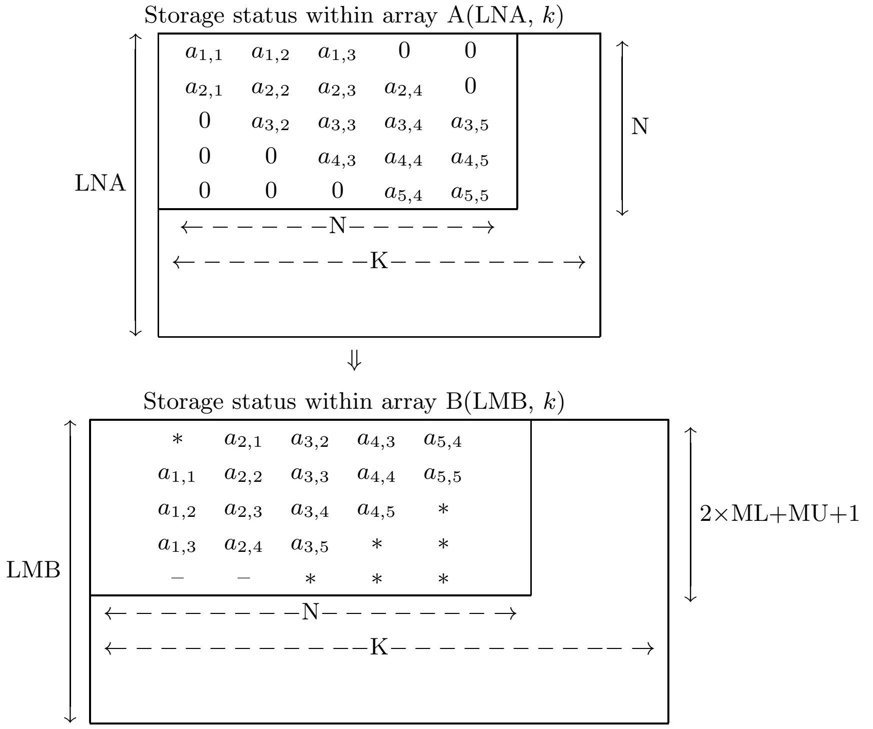
(d) $\text{MU} + \text{ML} < \text{LMB}$

(5) **Error indicator**

IERR value	Meaning	Processing
0	Normal termination.	
1000	N was equal to 1.	Processing continues.
3000	Restriction (a), (b), (c) or (d) was not satisfied.	Processing is aborted.

(6) **Notes**

(a) Array B elements, that were not corresponding to the elements of matrix A, retain the values they had at the time the subroutine was called.
Examples:


Storage status within array A(LNA, k)
$a_{1,1}$ $a_{1,2}$ $a_{1,3}$ 0 0
$a_{2,1}$ $a_{2,2}$ $a_{2,3}$ $a_{2,4}$ 0
0 $a_{3,2}$ $a_{3,3}$ $a_{3,4}$ $a_{3,5}$
0 0 $a_{4,3}$ $a_{4,4}$ $a_{4,5}$
0 0 0 $a_{5,4}$ $a_{5,5}$
LNA
N
N
K
⇓
Storage status within array B(LMB, k)
$*$ $a_{2,1}$ $a_{3,2}$ $a_{4,3}$ $a_{5,4}$
$a_{1,1}$ $a_{2,2}$ $a_{3,3}$ $a_{4,4}$ $a_{5,5}$
$a_{1,2}$ $a_{2,3}$ $a_{3,4}$ $a_{4,5}$ $*$
$a_{1,3}$ $a_{2,4}$ $a_{3,5}$ $*$ $*$
– – $*$ $*$ $*$
LMB
2×ML+MU+1
N
K

Remarks

a. Elements of B indicated by asterisks ($*$) and dashes (–) remain their input-time values.

b. The area indicated by dashes (–) is required for an LU decomposition of the matrix.

c. MU is the upper band width and ML is the lower band width.

d. LMB > ML + MU and K $\geq$ N must hold. (However, if LU decomposition is to be performed after conversion, LMB $\geq 2 \times$ ML + MU + 1 and K $\geq$ N must hold.)

(b) If an LU decomposition is to be performed after conversion, array B must be declared so that LMB satisfies the following condition.

$$\mathrm{LMB} \geq \min(2 \times \mathrm{ML} + \mathrm{MU} + 1, \mathrm{N} + \mathrm{ML})$$

(7) **Example**

Convert the storage mode of the real band matrix A from two-dimensional array type to band type and solve the simultaneous linear equations $A\boldsymbol{x} = \boldsymbol{b}$ using the array AC holding the matrix with converted mode (MU is the upper band width and ML is the lower band width).

```
IMPLICIT REAL(8) (A-H, O-Z)
PARAMETER (LNA=11, LMB=11)
DIMENSION A(LNA, LNA), AC(LMB, LMB), B(LNA), IPVT(LNA)
 ≀
CALL DABMCS(A, LNA, N, MU, ML, AC, LMB, IERR)
 ≀
CALL DBBDSL(AC, LMB, N, MU, ML, B, IPVT, JERR)
 ≀
```

2.2.2 DABMEL, RABMEL Storage Mode Conversion of a Real Band Matrix: from (Band Type) to (Two-Dimensional Array Type)

(1) **Function**

DABMEL or RABMEL converts the storage mode of the real band matrix A from (band type) to (two-dimensional array type).

(2) **Usage**

Double precision:
CALL DABMEL (A, LMA, N, MU, ML, B, LNB, IERR)
Single precision:
CALL RABMEL (A, LMA, N, MU, ML, B, LNB, IERR)

(3) **Arguments**

D:Double precision real Z:Double precision complex
R:Single precision real C:Single precision complex
I: $\left\{\begin{array}{l}\text{INTEGER(4) as for 32bit Integer}\\ \text{INTEGER(8) as for 64bit Integer}\end{array}\right\}$

No.	Argument	Type	Size	Input/ Output	Contents
1	A	$\left\{\begin{array}{c}\text{D}\\ \text{R}\end{array}\right\}$	LMA, N	Input	Real band matrix A (band type) (See Appendix B).
2	LMA	I	1	Input	Adjustable dimension of array A.
3	N	I	1	Input	Order of matrix A.
4	MU	I	1	Input	Upper band width of matrix A.
5	ML	I	1	Input	Lower band width of matrix A.
6	B	$\left\{\begin{array}{c}\text{D}\\ \text{R}\end{array}\right\}$	LNB, N	Output	Real band matrix A (two-dimensional array type) (See Appendix B).
7	LNB	I	1	Input	Adjustable dimension of array B.
8	IERR	I	1	Output	Error indicator

(4) **Restrictions**

(a) $0 \leq \text{MU} < \text{N}$

(b) $0 \leq \text{ML} < \text{N}$

(c) $0 < \text{N} \leq \text{LNB}$

(d) $\text{MU} + \text{ML} < \text{LMA}$

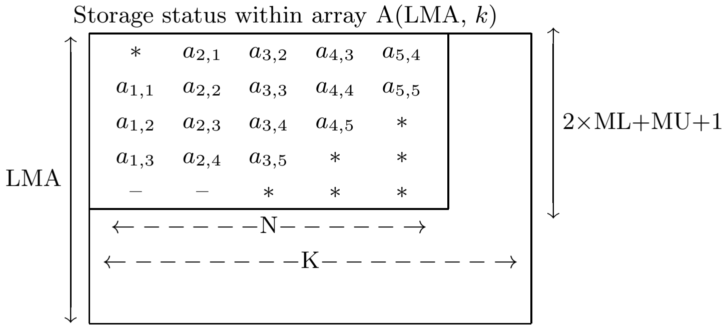
(5) **Error indicator**

IERR value	Meaning	Processing
0	Normal termination.	
1000	N was equal to 1.	Processing continues.
3000	Restriction (a), (b), (c) or (d) was not satisfied.	Processing is aborted.

(6) **Notes**

(a) 0.0 is entered in portions of the converted matrix that are outside of the band width.
Example:

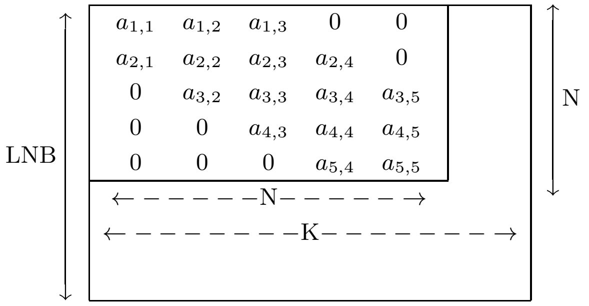
Storage status within array A(LMA, k)



$*$	$a_{2,1}$	$a_{3,2}$	$a_{4,3}$	$a_{5,4}$
$a_{1,1}$	$a_{2,2}$	$a_{3,3}$	$a_{4,4}$	$a_{5,5}$
$a_{1,2}$	$a_{2,3}$	$a_{3,4}$	$a_{4,5}$	$*$
$a_{1,3}$	$a_{2,4}$	$a_{3,5}$	$*$	$*$
$-$	$-$	$*$	$*$	$*$

⇓

Storage status within array B(LNB, k)



$a_{1,1}$	$a_{1,2}$	$a_{1,3}$	0	0
$a_{2,1}$	$a_{2,2}$	$a_{2,3}$	$a_{2,4}$	0
0	$a_{3,2}$	$a_{3,3}$	$a_{3,4}$	$a_{3,5}$
0	0	$a_{4,3}$	$a_{4,4}$	$a_{4,5}$
0	0	0	$a_{5,4}$	$a_{5,5}$

Remarks

a. The asterisk $*$ indicates an arbitrary value.

b. The area indicated by dashes ($-$) is required for an LU decomposition of the matrix.

c. MU is the upper band width and ML is the lower band width.

d. $\mathrm{LMA} > \mathrm{ML} + \mathrm{MU}$ and $\mathrm{K} \geq \mathrm{N}$ must hold.

(7) **Example**

Hold the real band matrix A in the array AC as band type, solve the simultaneous linear equation $A\boldsymbol{x} = \boldsymbol{b}$, and store LU decomposition of A in the array A as two-dimensional array type (MU is the upper band width and ML is the lower band width).

```
IMPLICIT REAL(8) (A-H, O-Z)
PARAMETER (LNA=11, LMB=11)
DIMENSION A(LNA, LNA), AC(LMB, LMB), B(LNA), IPVT(LNA)
 ≀
CALL DBBDSL(AC, LMB, N, MU, ML, B, IPVT, JERR)
 ≀
CALL DABMEL(AC, LMB, N, MU, ML, A, LNA, KERR)
 ≀
```

2.2.3 DASBCS, RASBCS Storage Mode Conversion of a Real Symmetric Band Matrix: from (Two-Dimensional Array Type) (Upper Triangular Type) to (Symmetric Band Type)

(1) **Function**

DASBCS or RASBCS converts the storage mode of the real symmetric band matrix A from (two-dimensional array type) to (symmetric band type).

(2) **Usage**

Double precision:
CALL DASBCS (A, LNA, N, MB, B, LMB, IERR)
Single precision:
CALL RASBCS (A, LNA, N, MB, B, LMB, IERR)

(3) **Arguments**

D:Double precision real Z:Double precision complex
R:Single precision real C:Single precision complex
I: $\left\{ \begin{array}{l} \text{INTEGER(4) as for 32bit Integer} \\ \text{INTEGER(8) as for 64bit Integer} \end{array} \right\}$

No.	Argument	Type	Size	Input/ Output	Contents
1	A	$\left\{\begin{array}{c}\text{D}\\\text{R}\end{array}\right\}$	LNA, N	Input	Real symmetric band matrix A (two-dimensional array type) (See Appendix B).
2	LNA	I	1	Input	Adjustable dimension of array A.
3	N	I	1	Input	Order of matrix A.
4	MB	I	1	Input	Band width of matrix A.
5	B	$\left\{\begin{array}{c}\text{D}\\\text{R}\end{array}\right\}$	LMB, N	Output	Real symmetric band matrix A (symmetric band type) (See Appendix B).
6	LMB	I	1	Input	Adjustable dimension of array B.
7	IERR	I	1	Output	Error indicator

(4) **Restrictions**

(a) $0 < \text{N} \leq \text{LNA}$

(b) $0 \leq \text{MB} < \text{N}$

(c) $\text{MB} < \text{LMB}$

(5) **Error indicator**

IERR value	Meaning	Processing
0	Normal termination.	
1000	N was equal to 1.	Processing continues.
3000	Restriction (a), (b) or (c) was not satisfied.	Processing is aborted.

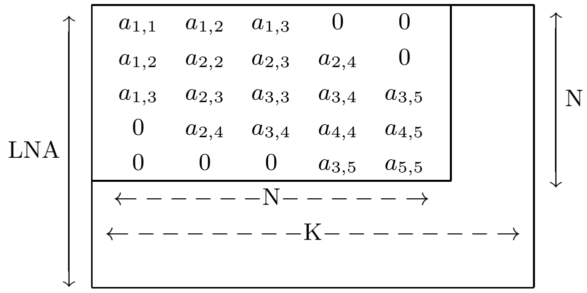
(6) **Notes**

(a) Only the upper triangular portion of matrix A is stored in array B.

(b) Array B elements, that were not corresponding to the elements of matrix A, retain the values they had at the time the subroutine was called.

Example:

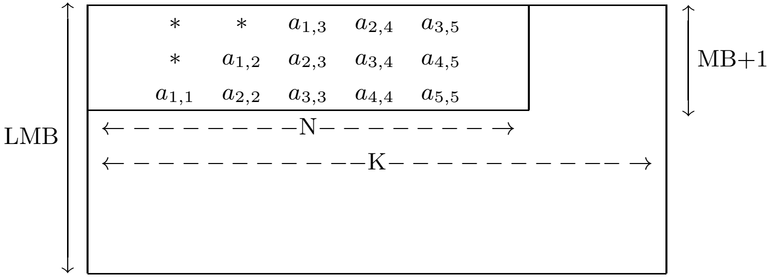
Storage status within array A(LNA, k)



$$\begin{array}{ccccc} a_{1,1} & a_{1,2} & a_{1,3} & 0 & 0 \\ a_{1,2} & a_{2,2} & a_{2,3} & a_{2,4} & 0 \\ a_{1,3} & a_{2,3} & a_{3,3} & a_{3,4} & a_{3,5} \\ 0 & a_{2,4} & a_{3,4} & a_{4,4} & a_{4,5} \\ 0 & 0 & 0 & a_{3,5} & a_{5,5} \end{array}$$

⇓

Storage status within array B(LMB, k)



$$\begin{array}{ccccc} * & * & a_{1,3} & a_{2,4} & a_{3,5} \\ * & a_{1,2} & a_{2,3} & a_{3,4} & a_{4,5} \\ a_{1,1} & a_{2,2} & a_{3,3} & a_{4,4} & a_{5,5} \end{array}$$

Remarks

a. Elements of B indicated be asterisks ($*$) retain their input-time values.

b. MB is the band width.

c. LMB $>$ MB and K $\geq$ N must hold.

(7) **Example**

Convert the storage mode of the positive symmetric band matrix A from two-dimensional array type to symmetric band type and solve the simultaneous linear equations $A\boldsymbol{x} = \boldsymbol{b}$ using the array AC holding the matrix with converted mode (MB is the band width).

```
IMPLICIT REAL(8) (A-H, O-Z)
PARAMETER (LNA=11, NC=11)
DIMENSION A(LNA, LNA), AC(LMB, LMB), B(NA)
  ≀
CALL DASBCS(A, LNA, N, MB, AC, LMB, IERR)
  ≀
CALL DBBPSL(AC, LMB, N, MB, B, JERR)
  ≀
```

2.2.4 DASBEL, RASBEL Storage Mode Conversion of a Real Symmetric Band Matrix: from (Symmetric Band Type) to (Two-Dimensional Array Type) (Upper Triangular Type)

(1) **Function**

DASBEL or RASBEL converts the storage mode of the real symmetric band matrix A from (symmetric band type) to (two-dimensional array type).

(2) **Usage**

Double precision:
CALL DASBEL (A, LMA, N, MB, B, LNB, IERR)
Single precision:
CALL RASBEL (A, LMA, N, MB, B, LNB, IERR)

(3) **Arguments**

D:Double precision real Z:Double precision complex I: $\left\{\begin{array}{l}\text{INTEGER(4) as for 32bit Integer}\\ \text{INTEGER(8) as for 64bit Integer}\end{array}\right\}$
R:Single precision real C:Single precision complex

No.	Argument	Type	Size	Input/ Output	Contents
1	A	$\left\{\begin{array}{c}D\\R\end{array}\right\}$	LMA, N	Input	Real symmetric band matrix A (symmetric band type) (See Appendix B).
2	LMA	I	1	Input	Adjustable dimension of array A.
3	N	I	1	Input	Order of matrix A.
4	MB	I	1	Input	Band width of matrix A.
5	B	$\left\{\begin{array}{c}D\\R\end{array}\right\}$	LNB, N	Output	Real symmetric band matrix A (two-dimensional array type) (See Appendix B).
6	LNB	I	1	Input	Adjustable dimension of array B.
7	IERR	I	1	Output	Error indicator

(4) **Restrictions**

(a) $0 < \mathrm{N} \leq \mathrm{LNB}$

(b) $0 \leq \mathrm{MB} < \mathrm{N}$

(c) $\mathrm{MB} < \mathrm{LMA}$

(5) **Error indicator**

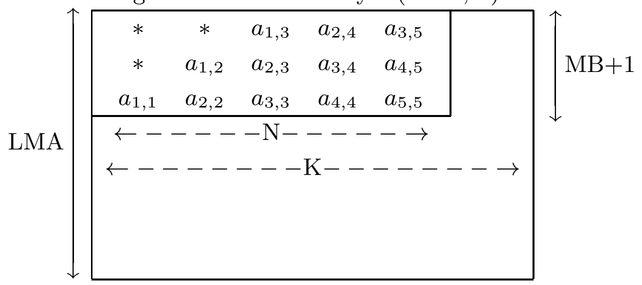
IERR value	Meaning	Processing
0	Normal termination.	
1000	N was equal to 1.	Processing continues.
3000	Restriction (a), (b) or (c) was not satisfied.	Processing is aborted.

(6) **Notes**

(a) The lower triangular portion is restored to the matrix after conversion. 0.0 is entered in portions of the converted matrix that are outside of the band width.

Example:

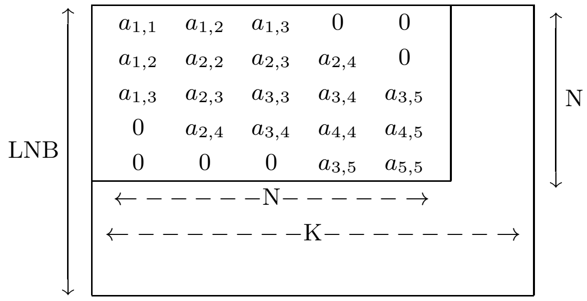
Storage status within array A(LMA, k)



$$\begin{array}{ccccc} * & * & a_{1,3} & a_{2,4} & a_{3,5} \\ * & a_{1,2} & a_{2,3} & a_{3,4} & a_{4,5} \\ a_{1,1} & a_{2,2} & a_{3,3} & a_{4,4} & a_{5,5} \end{array}$$

⇓

Storage status within array B(LNB, k)



$$\begin{array}{ccccc} a_{1,1} & a_{1,2} & a_{1,3} & 0 & 0 \\ a_{1,2} & a_{2,2} & a_{2,3} & a_{2,4} & 0 \\ a_{1,3} & a_{2,3} & a_{3,3} & a_{3,4} & a_{3,5} \\ 0 & a_{2,4} & a_{3,4} & a_{4,4} & a_{4,5} \\ 0 & 0 & 0 & a_{3,5} & a_{5,5} \end{array}$$

Remarks

a. The asterisk $*$ indicates an arbitrary value.

b. MB is the band width.

c. LMA $>$ MB, LNB $\geq$ N and K $\geq$ N must hold.

(7) **Example**

Hold the positive symmetric band matrix A in the array AC as symmetric band type, solve the simultaneous linear equation $A\boldsymbol{x} = \boldsymbol{b}$, and store LL^{T} decomposition of A in the array A as two-dimensional array type (MB is the band width).

```
IMPLICIT REAL(8) (A-H, O-Z)
PARAMETER (LNA=11, LMB=11)
DIMENSION A(LNA, LNA), AC(LMB, LMB), B(LNA)
  ≀
CALL DBBPSL(AC, LMB, N, MB, B, JERR)
  ≀
CALL DASBEL(AC, LMB, N, MB, A, LNA, KERR)
  ≀
```

2.2.5 DARSJD, RARSJD Storage Mode Conversion of a Real Symmetric Sparse Matrix: from (Real Symmetric One-Dimensional Row-Oriented List Type) (Upper Triangular Type) to (JAD)

(1) **Function**

DARSJD or RARSJD converts the storage mode of the real symmetric sparse matrix A from (real symmetric one-dimensional row-oriented list type) (upper triangular type) to (JAD; Jagged Diagonals Storage Type).

(2) **Usage**

Double precision:

CALL DARSJD (N, A, IA, JA, LXA, LXIA, MJAD, AJAD, IAJAD, JAJAD, JADORD, IW, IERR)

Single precision:

CALL RARSJD (N, A, IA, JA, LXA, LXIA, MJAD, AJAD, IAJAD, JAJAD, JADORD, IW, IERR)

(3) **Arguments**

D:Double precision real Z:Double precision complex I: $\left\{\begin{array}{l}\text{INTEGER(4) as for 32bit Integer}\\ \text{INTEGER(8) as for 64bit Integer}\end{array}\right\}$
R:Single precision real C:Single precision complex

No.	Argument	Type	Size	Input/ Output	Contents
1	N	I	1	Input	Order of matrix A.
2	A	$\left\{\begin{array}{c}D\\R\end{array}\right\}$	See Contents	Input	Real symmetric sparse matrix A (real symmetric one-dimensional row-oriented list type) (upper triangular type) **Size**: IA(N + 1) − 1 (See Appendix B).
3	IA	I	N+1	Input	Array of indices for sparse matrix A (real symmetric one-dimensional row-oriented list type) (upper triangular type) (See Appendix B).
4	JA	I	See Contents	Input	Array of indices for sparse matrix A (real symmetric one-dimensional row-oriented list type) (upper triangular type) **Size**: IA(N + 1) − 1 (See Appendix B).
5	LXA	I	1	Input	Size allocated for arrays AJAD and JAJAD.
6	LXIA	I	1	Input	Size allocated for array IAJAD.
7	MJAD	I	1	Output	Number of jagged diagonals for JAD storage of matrix A.
8	AJAD	$\left\{\begin{array}{c}D\\R\end{array}\right\}$	LXA	Output	Sparse matrix A (JAD storage type) (See Appendix B).

No.	Argument	Type	Size	Input/ Output	Contents
9	IAJAD	I	LXIA	Output	Array of indices for sparse matrix A (JAD storage type) (See Appendix B).
10	JAJAD	I	LXA	Output	Array of indices for sparse matrix A (JAD storage type) (See Appendix B).
11	JADORD	I	N	Output	Array of indices for sparse matrix A (JAD storage type) (See Appendix B).
12	IW	I	$3 \times \mathrm{N} + 1$	Work	Work area
13	IERR	I	1	Output	Error indicator

(4) **Restrictions**

(a) $\mathrm{N} > 0$

(b) $\mathrm{MJAD} \leq \mathrm{N}$

(c) $\mathrm{MJAD} < \mathrm{LXIA}$

(d) $\mathrm{IAJAD}(\mathrm{MJAD}+1) - 1 \leq \mathrm{LXA}$

(5) **Error indicator**

IERR value	Meaning	Processing
0	Normal termination.	
3000	Restriction (a) was not satisfied.	Processing is aborted.
3100	Restriction (b) was not satisfied. (Contradictions may exist among input value for N, A, IA, JA.)	
3200	Restriction (c) was not satisfied. (Size of array IAJAD for output is insufficient.)	
3300	Restriction (d) was not satisfied. (Either size of array AJAD or JAJAD is insufficient.)	

(6) **Notes**

(a) On input, only the upper nonzero elements of A must be stored in A, IA, and JA according to real symmetric one-dimensional row-oriented list type (upper triangular type). But on output, the whole nonzero elements, including the lower triangular ones, of A will be stored in AJAD, IAJAD, JAJAD, and JADORD according to JAD format.

(b) If there are some distinct rows that have the same number of nonzero elements, these rows will be placed in vertical series in JAD format. Naturally, any ordering among these rows is allowed for JAD format. Through this subroutine, these rows will be arranged on output so that a row which has a younger row number on input is placed lower than a row with an elder row number.

Example:

matrix$A$

$$\mathrm{N{=}7}\ \updownarrow\ \begin{array}{|ccccccc|}\hline 1.0 & 0.0 & 0.0 & 1.3 & 0.0 & 0.0 & 0.0 \\ 0.0 & 2.0 & 2.1 & 0.0 & 0.0 & 2.4 & 0.0 \\ 0.0 & 2.1 & 3.0 & 0.0 & 3.2 & 3.3 & 3.4 \\ 1.3 & 0.0 & 0.0 & 4.0 & 4.1 & 0.0 & 4.3 \\ 0.0 & 0.0 & 3.2 & 4.1 & 5.0 & 0.0 & 0.0 \\ 0.0 & 2.4 & 3.3 & 0.0 & 0.0 & 6.0 & 6.1 \\ 0.0 & 0.0 & 3.4 & 4.3 & 0.0 & 6.1 & 7.0 \\ \hline \end{array}$$

← − − − − − − − −N − − − − − − − − →

⇓

Storage status within array IA

1	3	6	10	13	14	16	17

← − − − − − (N + 1) − − − − − →

Storage status within array A

	1.0	1.3			(→ Leads to the left end of next row)
	2.0	2.1	2.4		(→ Leads to the left end of next row)
	3.0	3.2	3.3	3.4	(→ Leads to the left end of next row)
N	4.0	4.1	4.3		(→ Leads to the left end of next row)
	5.0				(→ Leads to the left end of next row)
	6.0	6.1			(→ Leads to the left end of next row)
	7.0				

=	1.0 1.3	2.0 2.1 2.4	3.0 3.2 3.3 3.4	4.0 4.1 4.3	5.0	6.0 6.1	7.0

← − − − − − − − − − − − − − − − − − − (IA(N + 1) − 1) − − − − − − − − − − − − − − − − − − →

Storage status within array JA

	1	4			(→ Leads to the left end of next row)
	2	3	6		(→ Leads to the left end of next row)
	3	5	6	7	(→ Leads to the left end of next row)
N	4	5	7		(→ Leads to the left end of next row)
	5				(→ Leads to the left end of next row)
	6	7			(→ Leads to the left end of next row)
	7				

=	1 4	2 3 6	3 5 6 7	4 5 7	5	6 7	7

← − − − − − − − − − − − − (IA(N + 1) − 1) − − − − − − − − − − − − →

⇓

Storage status within array JADORD

N	7
	6
	1
	4
	5
	3
	2

Storage status within array IAJAD

1	8	15	21	25	26

← −(MJAD + 1)− →

Storage status within array AJAD

← − − −MJAD − −− →

(a) ↓	(b) ↓	(c) ↓	(d) ↓	(e) ↓
2.1	3.0	3.2	3.3	3.4
3.4	4.3	6.1	7.0	
2.4	3.3	6.0	6.1	
1.3	4.0	4.1	4.3	
3.2	4.1	5.0	↓ (e)	
2.0	2.1	2.4		
1.0	1.3	↓ (d)		
↓ (b)	↓ (c)			

=

2.1 3.4 2.4 1.3 3.2 2.0 1.0	3.0 4.3 3.3 4.0 4.1 2.1 1.3	3.2 6.1 6.0 4.1 5.0 2.4	3.3 7.0 6.1 4.3	3.4

← − − − − − − − − − − − − − − − − − − −(IAJAD(N + 1) − 1) − − − − − − − − − − − − − − − − − →

Storage status within array JAJAD

← − − −MJAD − −− →

(a) ↓	(b) ↓	(c) ↓	(d) ↓	(e) ↓
2	3	5	6	7
3	4	6	7	
2	3	6	7	
1	4	5	7	
3	4	5	↓ (e)	
2	3	6		
1	4	↓ (d)		
↓ (b)	↓ (c)			

=

2 3 2 1 3 2 1	3 4 3 4 4 3 4	5 6 6 5 5 6	6 7 7 7	7

← − − − − − − − −(IAJAD(N + 1) − 1) − − − − − − −− →

Remarks

a. N is the order of matrix A.

b. To obtain JAD storage of matrix A, consider a data arrangement as follows:
Push rowwise the whole nonzero elements of matrix A to the left side, then sort the rows with respect to the number of nonzero elements in descending order;
The columns in this arrangement are called **jagged diagonal**s. The number of jagged diagonals is stored in the parameter MJAD. The elements are stored in array AJAD “jagged diagonal”wise, successively from the leftmost jagged diagonal to the rightmost one.

c. The row indices of the elements stored in array AJAD are stored in array JAJAD.

d. The indices of the starting element of each jagged diagonal in array AJAD are stored in IAJAD. The number of elements stored in AJAD added by 1, is stored in the MJAD + 1-th element of IAJAD.

e. The value 1 is set to IAJAD(1).

f. (The number of elements stored in JAD) = IAJAD(MJAD+1)−1.

(7) **Example**

Hold a real symmetric sparse matrix A in the array ACSR with real symmetric one-dimensional row-oriented list type (upper triangular type), convert the storage mode into JAD storage type, and then solve the eigenproblem $A\boldsymbol{x} = \lambda\boldsymbol{b}$ using the array AJAD holding the matrix with converted mode.

```
IMPLICIT REAL(8) (A-H, O-Z)
PARAMETER (N=7, NZ=11, LXA=NZ*2, LXIA=4)
DIMENSION ACSR(NZ),JACSR(NZ),IACSR(N+1)
DIMENSION AJAD(LXA),JACSR(LXA),IAJAD(LXIA),JADORD(N),IW(N*3+1)
  ≀
CALL DARSJD(N, ACSR, IACSR, JACSR, LXA, LXIA, &
            MJAD, AJAD, IAJAD, JAJAD, JADORD, IW, KERR)
  ≀
NA = IAJAD( MJAD + 1 ) - IAJAD(1)
CALL DCSJSS(MJAD, AJAD, NA, IAJAD, JAJAD, JADORD, &
            N, X, LDA, E, M, TR, IX, IS, ITM, IPREC, &
            NDIA, ITJD, ITQMR, IW, WK, IERR)
  ≀
```

2.2.6 DARGJM, RARGJM Storage Mode Conversion of a Sparse Matrix: from (Real One-Dimensional Row-Oriented Block List Type) to (MJAD; Multiple Jagged Diagonals Storage Type)

(1) **Function**

DARGJM or RARGJM converts the storage mode of real random sparse matrix A from (real one-dimensional row-oriented block list type) to (MJAD; multiple jagged diagonals storage type)

(2) **Usage**

Double precision:

CALL DARGJM (NB, M, A, IA, JA, ISW, LXA, LXIA, MJAD, AJAD, IAJAD, JAJAD, JADORD, IW, IERR)

Single precision:

CALL RARGJM (NB, M, A, IA, JA, ISW, LXA, LXIA, MJAD, AJAD, IAJAD, JAJAD, JADORD, IW, IERR)

(3) **Arguments**

D:Double precision real　Z:Double precision complex　I: $\left\{\begin{array}{l}\text{INTEGER(4) as for 32bit Integer}\\\text{INTEGER(8) as for 64bit Integer}\end{array}\right\}$
R:Single precision real　C:Single precision complex

No.	Argument	Type	Size	Input/ Output	Contents
1	NB	I	1	Input	Number of block rows (or columns) for dividing matrix A into M×M block matrix.
2	M	I	1	Input	Order of block.
3	A	$\left\{\begin{array}{c}\text{D}\\\text{R}\end{array}\right\}$	See Contents	Input	Random sparse matrix A (real row-oriented block list type) **Size**: (IA(NB+1)−IA(1))×M×M (See Appendix B).
4	IA	I	NB+1	Input	Array of indices for random sparse matrix A (real row-oriented block list type) (See Appendix B).
5	JA	I	See Contents	Input	Array of indices for random sparse matrix A (real row-oriented block list type) **Size**: IA(NB+1)−1 (See Appendix B).
6	ISW	I	1	Input	Processing Switch. ISW=0: Consecutive element in the row-oriented in block on memory. (Row Major) ISW=1: Consecutive element in the column-oriented in block on memory. (Column Major)
7	LXA	I	1	Input	Size allocated for arrays AJAD and JAJAD.
8	LXIA	I	1	Input	Size allocated for array IAJAD.
9	MJAD	I	1	Output	Number of jagged diagonals for MJAD storage of matrix A.

No.	Argument	Type	Size	Input/ Output	Contents
10	AJAD	$\begin{Bmatrix} \text{D} \\ \text{R} \end{Bmatrix}$	See Contents	Output	Sparse matrix A (MJAD storage type). **Size**: LXA×M×M (See Appendix B).
11	IAJAD	I	LXIA	Output	Array of indices for sparse matrix A (MJAD storage type) (See Appendix B).
12	JAJAD	I	LXA	Output	Array of indices for sparse matrix A (MJAD storage type) (See Appendix B).
13	JADORD	I	NB	Output	Array of indices for sparse matrix A (MJAD storage type) (See Appendix B).
14	IW	I	$2 \times \text{NB} + 1$	Work	Work area
15	IERR	I	1	Output	Error indicator.

(4) **Restrictions**

(a) NB > 0

(b) M > 0

(c) ISW $\in \{0, 1\}$

(d) MJAD $\leq$ NB

(e) MJAD $<$ LXIA

(f) IAJAD(MJAD $+ 1$) $-$ IAJAD(1) $\leq$ LXA

(5) **Error indicator**

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.	
3400	Restriction (e) was not satisfied. (Size of array IAJAD for output is insufficient.)	
3500	Restriction (f) was not satisfied. (Either size of array AJAD or JAJAD is insufficient.)	

(6) **Notes**

(a) Example of storage mode conversion of sparse matrix from real one-dimensional row-oriented block list type to MJAD storage type is described as follows.

Example:

Figure 2−1 Example of storage status of MJAD type. (For $\mathrm{M} = 2, \mathrm{ISW} = 0$)

matrix A

$$\mathrm{NB{=}4}\updownarrow \begin{bmatrix} B & 0 & C & 0 \\ D & F & 0 & G \\ 0 & 0 & H & 0 \\ 0 & K & 0 & L \end{bmatrix}$$

$$\leftarrow - - \mathrm{NB} - - \rightarrow$$

$$B = \begin{bmatrix} b_1 & b_2 \\ b_3 & b_4 \end{bmatrix},\quad C = \begin{bmatrix} c_1 & c_2 \\ c_3 & c_4 \end{bmatrix},\quad D = \begin{bmatrix} d_1 & d_2 \\ d_3 & d_4 \end{bmatrix},\quad F = \begin{bmatrix} f_1 & f_2 \\ f_3 & f_4 \end{bmatrix},\quad G = \underset{}{\overset{\leftarrow -\mathrm{M}- \rightarrow}{\begin{bmatrix} g_1 & g_2 \\ g_3 & g_4 \end{bmatrix}}},\ \updownarrow\mathrm{M}$$

$$H = \begin{bmatrix} h_1 & h_2 \\ h_3 & h_4 \end{bmatrix},\quad K = \begin{bmatrix} k_1 & k_2 \\ k_3 & k_4 \end{bmatrix},\quad L = \begin{bmatrix} l_1 & l_2 \\ l_3 & l_4 \end{bmatrix},\quad 0 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix},$$

⇓ Real one-dimensional row-oriented block list type Input

Storage status within array IA

1	3	6	7	9

$\leftarrow -(\mathrm{NB}+1)- \rightarrow$

Storage status within array A (For ISW = 0)

NB ↕

B	C		(→ Leads to the left end of next row)
D	F	G	(→ Leads to the left end of next row)
H			(→ Leads to the left end of next row)
K	L		

= | B C | D F G | H | K L |

$\leftarrow - - - - (\mathrm{IA}(\mathrm{NB}+1)-1) - - - - \rightarrow$

= | b_1 b_2 b_3 b_4 c_1 c_2 c_3 c_4 | $\cdots$ | k_1 k_2 k_3 k_4 l_1 l_2 l_3 l_4 |

$\leftarrow - - - - - - - - - - - ((\mathrm{IA}(\mathrm{NB}+1)-\mathrm{IA}(1)) \times \mathrm{M} \times \mathrm{M}) - - - - - - - - - - - \rightarrow$

Storage status within array JA

NB ↕

1	3		(→ Leads to the left end of next row)
1	2	4	(→ Leads to the left end of next row)
3			(→ Leads to the left end of next row)
2	4		

= | 1 3 | 1 2 4 | 3 | 2 4 |

$\leftarrow - - -(\mathrm{IA}(\mathrm{NB}+1)-1) - -- \rightarrow$

⇓ MJAD storage type Output

Storage status within array JADORD

NB ↕

3
1
4
2

Storage status within array IAJAD

1	5	8	9

$\leftarrow (\mathrm{MJAD}+1) \rightarrow$

(⋆)The order of block of jagged diagonal

$$\begin{array}{|c|c|c|}
\multicolumn{3}{c}{\leftarrow -\mathrm{MJAD}- \rightarrow}\\
\multicolumn{1}{c}{(a)} & \multicolumn{1}{c}{(b)} & \multicolumn{1}{c}{(c)}\\
\multicolumn{1}{c}{\downarrow} & \multicolumn{1}{c}{\downarrow} & \multicolumn{1}{c}{\downarrow}\\
\hline
D & F & G \\ \cline{3-3}
K & \multicolumn{1}{c|}{L} & \multicolumn{1}{c}{} \\
B & \multicolumn{1}{c|}{C} & \multicolumn{1}{c}{} \\ \cline{2-2}
H & \multicolumn{1}{c}{\downarrow} & \multicolumn{1}{c}{} \\ \cline{1-1}
\multicolumn{1}{c}{\downarrow} & \multicolumn{1}{c}{(c)} & \multicolumn{1}{c}{}\\
\multicolumn{1}{c}{(b)} & \multicolumn{1}{c}{} & \multicolumn{1}{c}{}
\end{array}$$

$$= \begin{array}{|cccc|ccc|c|}\hline D & K & B & H & F & L & C & G \\ \hline \end{array}$$

$\leftarrow (\mathrm{IAJAD}(\mathrm{MJAD}+1)-1) \rightarrow$

(⋆⋆)Storage status within array AJAD

$$\begin{array}{|c|c|c|c|c}
\multicolumn{4}{c}{\leftarrow - - \mathrm{M}\times\mathrm{M} - - \rightarrow} & \\
\multicolumn{1}{c}{(a)} & \multicolumn{1}{c}{(b)} & \multicolumn{1}{c}{(c)} & \multicolumn{1}{c}{(d)} & \\
\multicolumn{1}{c}{\downarrow} & \multicolumn{1}{c}{\downarrow} & \multicolumn{1}{c}{\downarrow} & \multicolumn{1}{c}{\downarrow} & \uparrow \\
\hline
d_1 & d_2 & d_3 & d_4 & \\
k_1 & k_2 & k_3 & k_4 & \\
b_1 & b_2 & b_3 & b_4 & \\
h_1 & h_2 & h_3 & h_4 & \mathrm{LXA} \\
f_1 & f_2 & f_3 & f_4 & \\
l_1 & l_2 & l_3 & l_4 & \\
c_1 & c_2 & c_3 & c_4 & \\
g_1 & g_2 & g_3 & g_4 & \downarrow \\
\hline
\multicolumn{1}{c}{\downarrow} & \multicolumn{1}{c}{\downarrow} & \multicolumn{1}{c}{\downarrow} & \multicolumn{1}{c}{} & \\
\multicolumn{1}{c}{(b)} & \multicolumn{1}{c}{(c)} & \multicolumn{1}{c}{(d)} & \multicolumn{1}{c}{} &
\end{array}$$

$$= \begin{array}{|cccccccc|c|cccccccc|}\hline d_1 & k_1 & b_1 & h_1 & f_1 & l_1 & c_1 & g_1 & \cdots & d_4 & k_4 & b_4 & h_4 & f_4 & l_4 & c_4 & g_4 \\ \hline \end{array}$$

$\leftarrow - - - - - -((\mathrm{IA}(\mathrm{NB}+1)-1)\times\mathrm{M}\times\mathrm{M}) - - - - - - \rightarrow$

Storage status within array JAJAD

$$\begin{array}{|c|c|c|}
\multicolumn{3}{c}{\leftarrow -\mathrm{MJAD}- \rightarrow}\\
\multicolumn{1}{c}{(a)} & \multicolumn{1}{c}{(b)} & \multicolumn{1}{c}{(c)}\\
\multicolumn{1}{c}{\downarrow} & \multicolumn{1}{c}{\downarrow} & \multicolumn{1}{c}{\downarrow}\\
\hline
1 & 2 & 4 \\ \cline{3-3}
2 & \multicolumn{1}{c|}{4} & \multicolumn{1}{c}{} \\
1 & \multicolumn{1}{c|}{3} & \multicolumn{1}{c}{} \\ \cline{2-2}
3 & \multicolumn{1}{c}{\downarrow} & \multicolumn{1}{c}{} \\ \cline{1-1}
\multicolumn{1}{c}{\downarrow} & \multicolumn{1}{c}{(c)} & \multicolumn{1}{c}{}\\
\multicolumn{1}{c}{(b)} & \multicolumn{1}{c}{} & \multicolumn{1}{c}{}
\end{array}$$

$$= \begin{array}{|cccc|ccc|c|}\hline 1 & 2 & 1 & 3 & 2 & 4 & 3 & 4 \\ \hline \end{array}$$

$\leftarrow - - - - - - - - - \rightarrow$

$(\mathrm{IAJAD}(\mathrm{MJAD}+1)-1)$

Remarks

a. NB is number of block rows (or columns) for dividing matrix A into M×M block matrix.

b. Push rowwise the whole nonzero block matrices of matrix A to the left side, then sort the rows with respect to the number of nonzero block matrix in descending order ($\star$);
The block columns in this arrangement are called jagged diagonals. The number of jagged diagonals is stored in the parameter MJAD. The storage method for array AJAD is described as follows. The first row, first column elements among from each block matrix (D, K, B, H, F, C, G) are taken. Here, these elements are arranged in the same order as block matrices appear along the jagged diagonal above $(d_1, k_1, b_1, h_1, f_1, c_1, g_1)$. This method perform repeatedly until M-th row, M-th column elements are taken((a), (b), (c), (d)) and stored in array AJAD.

c. The block column indices of the block array stored in array AJAD are stored in array JAJAD.

d. The indices of the starting element of each jagged diagonal in array AJAD stored in array IAJAD. The number of block array stored in AJAD added by 1, is stored in the MJAD+1-th element of AJAD.

e. The value 1 is set to IAJAD(1).

f. (The number of elements stored in MJAD) = (IAJAD(MJAD+1)−1)×M×M

g. If there are some distinct block rows that have the same number of nonzero block matrix, these block rows will be placed in vertical series in MJAD format. Naturally, any ordering among these block rows is allowed for MJAD format. Through this subroutine, these block rows will be arranged on output so that a block rows which has a younger block rows number on input is placed lower than a block rows with an elder block rows number.

h. Each elements in the same block will be arranged with equal intervals of LXA in memory ($\star\star$). For example, elements in the block D (d_1, d_2, d_3, d_4) will be arranged with equal intervals of LXA in memory.

i. For each elements in block of array A stored sequentially in column-oriented, you should be set 1 to ISW.

(b) Number of times of storage type conversion using this subroutine should be reduced if possible. For example, when you will calculate repeatedly matrix-vector products without changing matrix A for iterative solution methods of simultaneous linear equation, eigenvalue equation of sparse matrix and so on, calculation will be performed efficiently if you perform storage mode conversion only once using this subroutine outside the iteration loop and use repeatedly the matrix-vector products inside the iteration loop.

(c) When you want to calculate sparse matrix-vector products of MJAD type obtained by this subroutine, if the order of block $M = 1, 3$ or 4, you can calculate by using 3.2.24 $\left\{\begin{matrix}\text{DAMVJ1}\\ \text{RAMVJ1}\end{matrix}\right\}$, 3.2.25 $\left\{\begin{matrix}\text{DAMVJ3}\\ \text{RAMVJ3}\end{matrix}\right\}$ and 3.2.26 $\left\{\begin{matrix}\text{DAMVJ4}\\ \text{RAMVJ4}\end{matrix}\right\}$, respectively. These subroutines will calculate efficiently because of enhanced assembly tuning for Vector Engine.

(7) **Example**

Hold the random sparse matrix A that have element of 3×3 block matrix in the array A as real one-dimensional row-oriented block list type, convert the storage mode into MJAD storage type, and then solve the matrix-vector conducts $\boldsymbol{y} = \beta\boldsymbol{y} + \alpha A\boldsymbol{x}$ using by the array AJAD holding the matrix with converted mode.

```
! *** EXAMPLE OF DARGJM AND DAMVJ1 ***
      INTEGER NB,NZ,LXA,LXIA,M
      PARAMETER( NB=4, M=3, NZ=8, ISW=0, LXA=NZ, LXIA=N+1 )
      INTEGER IA(NB+1),JA(LXA),MJAD,IAJAD(LXIA),JAJAD(LXA),JADORD(NB)
      INTEGER IW(NB*2+1),IERR
      INTEGER I,J,K,L
      REAL(8) A(NZ*M*M),AJAD(LXA*M*M),X(NB*M),Y(NB*M),W(NB*M)
      REAL(8) ALPHA,BETA
      PARAMETER (ALPHA=1.0D0, BETA=1.0D0)
       ≀
      CALL DARGJM &
      (NB,M,A,IA,JA,ISW,LXA,LXIA,MJAD,AJAD,IAJAD,JAJAD,JADORD,IW,IERR)
       ≀
```

```
CALL DAMVJ3 &
(AJAD,LXA,IAJAD,JAJAD,JADORD,NB,MJAD,ALPHA,BETA,X,Y,W,IERR)
 ≀
```

2.2.7 ZARSJD, CARSJD Storage Mode Conversion of a Hermitian Sparse Matrix: from (Hermitian One-Dimensional Row-Oriented List Type) (Upper Triangular Type) to (JAD; Jagged Diagonals Storage Type)

(1) **Function**

ZARSJD or CARSJD converts the storage mode of the complex Hermitian sparse matrix A from (Hermitian one-dimensional row-oriented list type)(upper triangular type) to (JAD; jagged diagonals storage type).

(2) **Usage**

Double precision:

CALL ZARSJD (N, A, IA, JA, LXA, LXIA, MJAD, AJAD, IAJAD, JAJAD, JADORD, IW, IERR)

Single precision:

CALL CARSJD (N, A, IA, JA, LXA, LXIA, MJAD, AJAD, IAJAD, JAJAD, JADORD, IW, IERR)

(3) **Arguments**

D:Double precision real Z:Double precision complex I: $\left\{\begin{array}{l}\text{INTEGER(4) as for 32bit Integer}\\\text{INTEGER(8) as for 64bit Integer}\end{array}\right\}$
R:Single precision real C:Single precision complex

No.	Argument	Type	Size	Input/ Output	Contents
1	N	I	1	Input	Order of matrix A
2	A	$\left\{\begin{matrix}\text{Z}\\\text{C}\end{matrix}\right\}$	See Contents	Input	Hermitian sparse matrix A (Hermitian one-dimensional row-oriented list type) (upper triangular type) **Size**: IA(N + 1) − 1 (See Appendix B).
3	IA	I	N + 1	Input	Array of indices for sparse matrix A (Hermitian one-dimensional row-oriented list type) (upper triangular type) (See Appendix B).
4	JA	I	See Contents	Input	Array of indices for sparse matrix A (Hermitian one-dimensional row-oriented list type) (upper triangular type) **Size**: IA(N + 1) − 1 (See Appendix B).
5	LXA	I	1	Input	Size allocated for arrays AJAD and JAJAD.
6	LXIA	I	1	Input	Size allocated for array IAJAD.
7	MJAD	I	1	Output	Number of jagged diagonals for JAD storage of matrix A.
8	AJAD	$\left\{\begin{matrix}\text{Z}\\\text{C}\end{matrix}\right\}$	LXA	Output	Sparse matrix A (JAD storage type) (See Appendix B).
9	IAJAD	I	LXIA	Output	Array of indices for sparse matrix A (JAD storage type) (See Appendix B).

No.	Argument	Type	Size	Input/ Output	Contents
10	JAJAD	I	LXA	Output	Array of indices for sparse matrix A (JAD storage type) (See Appendix B).
11	JADORD	I	N	Output	Array of indices for sparse matrix A (JAD storage type) (See Appendix B).
12	IW	I	$3 \times \mathrm{N} + 1$	Work	Work area
13	IERR	I	1	Output	Error indicator

(4) **Restrictions**

(a) $\mathrm{N} > 0$

(b) $\mathrm{MJAD} \leq \mathrm{N}$

(c) $\mathrm{MJAD} < \mathrm{LXIA}$

(d) $\mathrm{IAJAD(MJAD + 1) - 1} \leq \mathrm{LXA}$

(5) **Error indicator**

IERR value	Meaning	Processing
0	Normal termination.	
3000	Restriction (a) was not satisfied.	Processing is aborted.
3100	Restriction (b) was not satisfied. (Contradictions may exist among input value for N, A, IA, JA.)	
3200	Restriction (c) was not satisfied. (Size of array IAJAD for output is insufficient.)	
3300	Restriction (d) was not satisfied. (Either size of array AJAD or JAJAD is insufficient.)	

(6) **Notes**

(a) On input, only the upper nonzero elements of A must be stored in A, IA, and JA according to Hermitian one-dimensional row-oriented list type (upper triangular type). But on output, the whole nonzero elements, including the lower triangular ones, of A will be stored in AJAD, IAJAD, JAJAD, and JADORD according to JAD format.

(b) If there are some distinct rows that have the same number of nonzero elements, these rows will be placed in vertical series in JAD format. Naturally, any ordering among these rows is allowed for JAD format. Through this subroutine, these rows will be arranged on output so that a row which has a younger row number on input is placed lower than a row with an elder row number.

Example:

matrix$A$

$$\text{N=7}\ \updownarrow \begin{array}{|ccccccc|} \hline a_{1,1} & 0. & 0. & a_{1,4} & 0. & 0. & 0. \\ 0. & a_{2,2} & a_{2,3} & 0. & 0. & a_{2,6} & 0. \\ 0. & \overline{a}_{2,3} & a_{3,3} & 0. & a_{3,5} & a_{3,6} & a_{3,7} \\ \overline{a}_{1,4} & 0. & 0. & a_{4,4} & a_{4,5} & 0. & a_{4,7} \\ 0. & 0. & \overline{a}_{3,5} & \overline{a}_{4,5} & a_{5,5} & 0. & 0. \\ 0. & \overline{a}_{2,6} & \overline{a}_{3,6} & 0. & 0. & a_{6,6} & a_{6,7} \\ 0. & 0. & \overline{a}_{3,7} & \overline{a}_{4,7} & 0. & \overline{a}_{6,7} & a_{7,7} \\ \hline \end{array}$$

← − − − − − − − −N − − − − − − − − →

⇓

Storage status within array IA

1	3	6	10	13	14	16	17

← − − − − − (N + 1) − − − − − →

Storage status within array A

	$a_{1,1}$	$a_{1,4}$			(→ Leads to the left end of next row)
	$a_{2,2}$	$a_{2,3}$	$a_{2,6}$		(→ Leads to the left end of next row)
	$a_{3,3}$	$a_{3,5}$	$a_{3,6}$	$a_{3,7}$	(→ Leads to the left end of next row)
N	$a_{4,4}$	$a_{4,5}$	$a_{4,7}$		(→ Leads to the left end of next row)
	$a_{5,5}$				(→ Leads to the left end of next row)
	$a_{6,6}$	$a_{6,7}$			(→ Leads to the left end of next row)
	$a_{7,7}$				

=	$a_{1,1}$ $a_{1,4}$	$a_{2,2}$ $a_{2,3}$ $a_{2,6}$	$a_{3,3}$ $a_{3,5}$ $a_{3,6}$ $a_{3,7}$	$a_{4,4}$ $a_{4,5}$ $a_{4,7}$	$a_{5,5}$	$a_{6,6}$ $a_{6,7}$	$a_{7,7}$

← − (IA(N + 1) − 1) − →

Storage status within array JA

	1	4			(→ Leads to the left end of next row)
	2	3	6		(→ Leads to the left end of next row)
	3	5	6	7	(→ Leads to the left end of next row)
N	4	5	7		(→ Leads to the left end of next row)
	5				(→ Leads to the left end of next row)
	6	7			(→ Leads to the left end of next row)
	7				

=	1 4	2 3 6	3 5 6 7	4 5 7	5	6 7	7

← − − − − − − − − − − − (IA(N + 1) − 1) − − − − − − − − − − − →

⇓

Storage status within array JADORD

N ↕
7
6
1
4
5
3
2

Storage status within array IAJAD

1	8	15	21	25	26

← −(MJAD + 1)− →

Storage status within array AJAD

← − − −MJAD − − − →

(a) ↓	(b) ↓	(c) ↓	(d) ↓	(e) ↓
$\overline{a}_{2,3}$	$a_{3,3}$	$a_{3,5}$	$a_{3,6}$	$a_{3,7}$
$\overline{a}_{3,7}$	$\overline{a}_{4,7}$	$\overline{a}_{6,7}$	$a_{7,7}$	
$\overline{a}_{2,6}$	$\overline{a}_{3,6}$	$a_{6,6}$	$a_{6,7}$	
$\overline{a}_{1,4}$	$a_{4,4}$	$a_{4,5}$	$a_{4,7}$	
$\overline{a}_{3,5}$	$\overline{a}_{4,5}$	$a_{5,5}$	↓	
$a_{2,2}$	$a_{2,3}$	$a_{2,6}$	(e)	
$a_{1,1}$	$a_{1,4}$	↓		
↓	↓	(d)		
(b)	(c)			

= | $\overline{a}_{2,3}$ $\overline{a}_{3,7}$ $\overline{a}_{2,6}$ $\overline{a}_{1,4}$ $\overline{a}_{3,5}$ $a_{2,2}$ $a_{1,1}$ | $a_{3,3}$ $\overline{a}_{4,7}$ $\overline{a}_{3,6}$ $a_{4,4}$ $\overline{a}_{4,5}$ $a_{2,3}$ $a_{1,4}$ | $a_{3,5}$ $\overline{a}_{6,7}$ $a_{6,6}$ $a_{4,5}$ $a_{5,5}$ $a_{2,6}$ | $a_{3,6}$ $a_{7,7}$ $a_{6,7}$ $a_{4,7}$ | $a_{3,7}$ |

← −(IAJAD(N + 1) − 1) − →

Storage status within array JAJAD

← − − −MJAD − − − →

(a) ↓	(b) ↓	(c) ↓	(d) ↓	(e) ↓
2	3	5	6	7
3	4	6	7	
2	3	6	7	
1	4	5	7	
3	4	5	↓	
2	3	6	(e)	
1	4	↓		
↓	↓	(d)		
(b)	(c)			

= | 2 3 2 1 3 2 1 | 3 4 3 4 4 3 4 | 5 6 6 5 5 6 | 6 7 7 7 | 7 |

← − − − − − − − −(IAJAD(N + 1) − 1) − − − − − − − − →

Remarks

a. The $\overline{x}$ indicates the complex conjugate of the x.

b. N is the order of matrix A.

c. To obtain JAD storage of matrix A, consider a data arrangement as follows:
Push rowwise the whole nonzero elements of matrix A to the left side, then sort the rows with respect to the number of nonzero elements in descending order;
The columns in this arrangement are called **jagged diagonal**s. The number of jagged diagonals is stored in the parameter MJAD. The elements are stored in array AJAD "jagged diagonal"wise, successively from the leftmost jagged diagonal to the rightmost one.

d. The row indices of the elements stored in array AJAD are stored in array JAJAD.

e. The indices of the starting element of each jagged diagonal in array AJAD are stored in IAJAD. The number of elements stored in AJAD added by 1, is stored in the MJAD + 1-th element of IAJAD.

f. The value 1 is set to IAJAD(1).

g. (The number of elements stored in JAD) = IAJAD(MJAD+1)−1.

(7) **Example**

Hold a Hermitian sparse matrix A in the array ACSR with Hermitian one-dimensional row-oriented list type (upper triangular type), convert the storage mode into JAD storage type, and then solve the eigenproblem $A\boldsymbol{x} = \lambda\boldsymbol{b}$ using the array AJAD holding the matrix with converted mode.

```
IMPLICIT COMPLEX(8) (A-H, O-Z)
PARAMETER (N=7, NZ=11, LXA=NZ*2, LXIA=4)
DIMENSION ACSR(NZ),JACSR(NZ),IACSR(N+1)
DIMENSION AJAD(LXA),JACSR(LXA),IAJAD(LXIA),JADORD(N),IW(N*3+1)
  ≀
CALL ZARSJD(N, ACSR, IACSR, JACSR, LXA, LXIA, &
            MJAD, AJAD, IAJAD, JAJAD, JADORD, IW, KERR)
  ≀
NA = IAJAD( MJAD + 1 ) - IAJAD(1)
CALL ZCHJSS(MJAD, AJAD, NA, IAJAD, JAJAD, JADORD, &
            N, X, LDA, E, M, TR, IX, IS, ITM, IPREC, &
            NDIA, ITJD, ITQMR, IW, WK, IERR)
  ≀
```

2.2.8 ZARGJM, CARGJM Storage Mode Conversion of a Sparse Matrix: from (Complex One-Dimensional Row-Oriented Block List Type) to (MJAD; Multiple Jagged Diagonals Storage Type)

(1) **Function**

ZARGJM or CARGJM converts the storage mode of complex random sparse matrix A from (complex one-dimensional row-oriented block list type) to (MJAD; multiple jagged diagonals storage type)

(2) **Usage**

Double precision:

CALL ZARGJM (NB, M, A, IA, JA, ISW, LXA, LXIA, MJAD, AJAD, IAJAD, JAJAD, JADORD, IW, IERR)

Single precision:

CALL CARGJM (NB, M, A, IA, JA, ISW, LXA, LXIA, MJAD, AJAD, IAJAD, JAJAD, JADORD, IW, IERR)

(3) **Arguments**

D:Double precision real Z:Double precision complex I: { INTEGER(4) as for 32bit Integer
R:Single precision real C:Single precision complex INTEGER(8) as for 64bit Integer }

No.	Argument	Type	Size	Input/ Output	Contents
1	NB	I	1	Input	Number of block rows (or columns) for dividing matrix A into M×M block matrix.
2	M	I	1	Input	Order of block.
3	A	{Z C}	See Contents	Input	Random sparse matrix A (complex row-oriented block list type) **Size**: (IA(NB+1)−IA(1))×M×M (See Appendix B).
4	IA	I	NB+1	Input	Array of indices for random sparse matrix A (complex row-oriented block list type) (See Appendix B).
5	JA	I	See Contents	Input	Array of indices for random sparse matrix A (complex row-oriented block list type) **Size**: IA(NB+1)−1 (See Appendix B).
6	ISW	I	1	Input	Processing Switch. ISW=0: Consecutive element in the row-oriented in block on memory. (Row Major) ISW=1: Consecutive element in the column-oriented in block on memory. (Column Major)
7	LXA	I	1	Input	Size allocated for arrays AJAD and JAJAD.
8	LXIA	I	1	Input	Size allocated for array IAJAD.
9	MJAD	I	1	Output	Number of jagged diagonals for MJAD storage of matrix A.

No.	Argument	Type	Size	Input/ Output	Contents
10	AJAD	$\begin{Bmatrix} \mathrm{Z} \\ \mathrm{C} \end{Bmatrix}$	See Contents	Output	Sparse matrix A (MJAD storage type). **Size**: LXA×M×M (See Appendix B)
11	IAJAD	I	LXIA	Output	Array of indices for sparse matrix A (MJAD storage type) (See Appendix B).
12	JAJAD	I	LXA	Output	Array of indices for sparse matrix A (MJAD storage type) (See Appendix B).
13	JADORD	I	NB	Output	Array of indices for sparse matrix A (MJAD storage type) (See Appendix B).
14	IW	I	2×NB+1	Work	Work area
15	IERR	I	1	Output	Error indicator.

(4) **Restrictions**

(a) $\mathrm{NB} > 0$

(b) $\mathrm{M} > 0$

(c) $\mathrm{ISW} \in \{0, 1\}$

(d) $\mathrm{MJAD} \leq \mathrm{NB}$

(e) $\mathrm{MJAD} < \mathrm{LXIA}$

(f) $\mathrm{IAJAD(MJAD + 1) - IAJAD(1)} \leq \mathrm{LXA}$

(5) **Error indicator**

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.	
3400	Restriction (e) was not satisfied. (Size of array IAJAD for output is insufficient.)	
3500	Restriction (f) was not satisfied. (Either size of array AJAD or JAJAD is insufficient.)	

(6) **Notes**

(a) Example of storage mode conversion of sparse matrix is shown in 2.2.6 figure 2−1.

(b) Number of times of storage type conversion using this subroutine should be reduced if possible. For example, when you will calculate repeatedly matrix-vector products without changing matrix A for iterative solution methods of simultaneous linear equation, eigenvalue equation of sparse matrix and so on, calculation will be performed efficiently if you perform storage mode conversion only once using this subroutine outside the iteration loop and use repeatedly the matrix-vector products inside the iteration loop.

Remarks

a. If there are some distinct block rows that have the same number of nonzero block matrix, these block rows will be placed in vertical series in MJAD format. Naturally, any ordering among these block rows is allowed for MJAD format. Through this subroutine, these block rows will be arranged on output so that a block rows which has a younger block rows number on input is placed lower than a block rows with an elder block rows number.

b. Each elements in the same block will be arranged with equal intervals of LXA in memory (⋆⋆). For example, elements in the block D (d_1, d_2, d_3, d_4) will be arranged with equal intervals of LXA in memory. (See Note (2.2.6 figure 2−1)).

(c) When you want to calculate sparse matrix-vector products of MJAD type obtained by this subroutine, if the order of block $M = 1$, you can calculate by using 3.2.27 $\left\{\begin{matrix}\text{ZANVJ1}\\ \text{CANVJ1}\end{matrix}\right\}$. These subroutines will calculate efficiently because of enhanced assembly tuning for Vector Engine.

(7) **Example**

Hold the random sparse matrix A in the array A as complex one-dimensional row-oriented block list type, convert the storage mode into MJAD storage type, and then solve the matrix-vector conducts $\boldsymbol{y} = \beta\boldsymbol{y} + \alpha A\boldsymbol{x}$ using by the array AJAD holding the matrix with converted mode.

```
! *** EXAMPLE OF ZARGJM AND ZANVJ1 ***
      INTEGER NB,NZ,LXA,LXIA,M
      PARAMETER( NB=4, M=1, NZ=8, ISW=0, LXA=NZ, LXIA=N+1 )
      INTEGER IA(NB+1),JA(LXA),MJAD,IAJAD(LXIA),JAJAD(LXA),JADORD(NB)
      INTEGER IW(NB*2+1),IERR
      INTEGER I,J,K,L
      COMPLEX(8) A(NZ*M*M),AJAD(LXA*M*M),X(NB*M),Y(NB*M),W(NB*M)
      COMPLEX(8) ALPHA,BETA
      PARAMETER (ALPHA=(1.0D0,0.D0), BETA=(1.0D0,0.D0) )
       ≀
      CALL ZARGJM &
      (NB,M,A,IA,JA,ISW,LXA,LXIA,MJAD,AJAD,IAJAD,JAJAD,JADORD,IW,IERR)
       ≀
      CALL ZANVJ1 &
      (AJAD,LXA,IAJAD,JAJAD,JADORD,NB,MJAD,ALPHA,BETA,X,Y,W,IERR)
       ≀
```

2.2.9 DXA005, RXA005 Storage Mode Conversion of the Sparse Matrix : from (One-Dimensional Column-Oriented List Format) to (ELLPACK Format)

(1) **Function**

DXA005 or RXA005 converts the storage mode of the sparse matrix A from (One-dimensional Column-oriented List Format) to (ELLPACK Format).

(2) **Usage**

Double precision:

CALL DXA005 (ITYPE, N, VALUES, IPONTR, IRWIND, NNZ, A, LNA, M, JA, IWK, IERR)

Single precision:

CALL RXA005 (ITYPE, N, VALUES, IPONTR, IRWIND, NNZ, A, LNA, M, JA, IWK, IERR)

(3) **Arguments**

D:Double precision real	Z:Double precision complex	I: {INTEGER(4) as for 32bit Integer}
R:Single precision real	C:Single precision complex	

No.	Argument	Type	Size	Input/ Output	Contents
1	ITYPE	I	1	Input	Switch to specify how the matrix data is to be referenced for Column Oriented One-Dimensional List Type ITYPE= 1 : Both upper and lower triangular parts (Asymmetric case) 2 : Either upper or lower triangular parts (Symmetric case)
2	N	I	1	Input	Order of the matrix A
3	VALUES	$\begin{Bmatrix} \text{D} \\ \text{R} \end{Bmatrix}$	NNZ	Input	Nonzero element values of the matrix A (One-dimensional Column-oriented List Format) (See Appendix B)
4	IPONTR	I	N+1	Input	The indices within IRWIND that indicate the positions of the first elements in each column. As for Column j in which there are no elements to input, IPONTR should be determined so that : $\mathrm{IPONTR(j)} = \mathrm{IPONTR(j+1)}$ $(\mathrm{j} = 1, \cdots, \mathrm{N})$
5	IRWIND	I	NNZ	Input	Row indices of nonzero elements in Matrix A Row indices of those elements of Matrix A whose values are stored in VALUES(i) should be stored in IRWIND(i). $(i = 1, \cdots, \mathrm{NNZ})$
6	NNZ	I	1	Input	The number of those elements in A whose values are stored in Array VALUES

No.	Argument	Type	Size	Input/ Output	Contents
7	A	$\left\{\begin{matrix}\text{D}\\\text{R}\end{matrix}\right\}$	LNA, M	Output	Array of nonzero element values of the matrix A
8	LNA	I	1	Input	Adjustable dimension of array A and JA
9	M	I	1	Input	M denotes the column number of both Array A and JA.
				Output	A size that M should have as its value at least (when IERR= 2000 is returned)
10	JA	I	LNA, M	Output	Array storing the nonzero structure data of the matrix A
11	IWK	I	N	Work	Work area
12	IERR	I	1	Output	Error indicator

(4) **Restrictions**

(a) $\text{ITYPE} \in \{1, 2\}$

(b) $1 \leq \text{N} \leq \text{LNA}$

(c) $\text{IPONTR}(1) = 1$
$1 \leq \text{IPONTR}(\text{j}) \leq \text{NNZ} + 1 \quad (\text{j} = 2, \ldots, \text{N})$
$\text{IPONTR}(\text{N} + 1) = \text{NNZ} + 1$

(d) $0 \leq \text{IPONTR}(\text{j} + 1) - \text{IPONTR}(\text{j}) \leq \text{N} \quad (\text{i} = 1, \ldots, \text{N})$

(e) $1 \leq \text{IRWIND}(\text{k}) \leq \text{N} \quad (\text{k} = 1, \ldots, \text{NNZ})$

(f) The values $\text{IRWIND}(\text{k}) \quad (\text{k} = \text{IPONTR}(\text{j}), \ldots, \text{IPONTR}(\text{j} + 1) - 1)$ should be distinct among all the elements that have the same column index j.

(g) $\text{NNZ} \geq 1$

(h) $\text{M} \geq 1$

(5) **Error indicator**

IERR value	Meaning	Processing
0	Normal termination.	
2000	The value M is too small.	Procedure is terminated after setting an output value to M, which should have been required as the size M.
2200	Some or all of the diagonal elements were not given as input data.	Procedure is continued assuming that all of those omitted diagonals have value 0.0.
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.	
3050	Restriction (e) was not satisfied.	
3060	Restriction (f) was not satisfied.	
3070	Restriction (g) was not satisfied.	
3080	Restriction (h) was not satisfied.	

(6) **Notes**

(a) See Appendix B to find how to set values to Arrays VALUES, IPONTR, IRWIND, A and JA.

(b) Procedure will be terminated without performing data transformation between the storage modes, if the input value M was insufficient (IERR = 2000). In that case, data transformation might be performed without fail in the successive call, by deallocating the already allocated memory once, and then by allocating memory of the size that was indicated as an output value M.

```
 ≀
CALL DXA005(ITYPE, N, VALUES, IPONTR, IRWIND, A, LNA, M, JA, IWK, IERR)
IF( IERR .EQ. 2000 ) THEN
   DEALLOCATE(A, JA)
   ALLOCATE (A(LNA, M), JA(LNA, M))
   CALL DXA005(ITYPE, N, VALUES, IPONTR, IRWIND, A, LNA, M, JA, IWK, IERR)
ENDIF
 ≀
```

(7) **Example**

(a) Problem
Convert the storage mode of the matrix A from one-dimensional column-oriented list format to ELLPACK format

$$A = \begin{bmatrix} 1.0 & 1.1 & 0.0 & 1.2 & 1.4 \\ -2.1 & 2.0 & 2.1 & 0.0 & 0.0 \\ 0.0 & 0.0 & 3.0 & 0.0 & 3.5 \\ 4.5 & 0.0 & 0.0 & 4.0 & 4.1 \\ 5.0 & 0.0 & -5.4 & -5.1 & 5.0 \end{bmatrix}$$

(b) Input data
Values of matrix elements, locations of matrix elements, column indices of matrix elements, ITYPE = 1, N = 5, LNA = 5, M = 1

(c) Main program

```
      PROGRAM BXA005
! *** EXAMPLE OF DXA005 ***
      IMPLICIT NONE
!
      INTEGER N,NNZ,LNA
      PARAMETER( N = 5, NNZ = 16, LNA = 5 )
      INTEGER ITYPE,IPONTR(N+1),IRWIND(NNZ),M,IWK(N),IERR
      REAL(8) VALUES(NNZ)
      INTEGER,ALLOCATABLE :: JA(:,:)
      REAL(8),ALLOCATABLE :: A(:,:)
!
      INTEGER I,J,KERR
!
      KERR  = 0
      ITYPE = 1
!
      DO 100 I= 1,NNZ
         READ(5,*) VALUES(I)
  100 CONTINUE
      DO 110 I= 1,N+1
         READ(5,*) IPONTR(I)
  110 CONTINUE
      DO 120 I= 1,NNZ
         READ(5,*) IRWIND(I)
  120 CONTINUE
!
      WRITE(6,6000) N,NNZ,LNA
      WRITE(6,6010)
      DO 130 I=1,NNZ
         WRITE(6,6020) VALUES(I)
  130 CONTINUE
      WRITE(6,6030)
      DO 140 I=1,N+1
```

```
         WRITE(6,6040) IPONTR(I)
  140 CONTINUE
      WRITE(6,6050)
      DO 150 I=1,NNZ
         WRITE(6,6040) IRWIND(I)
  150 CONTINUE
!
      M=1
      WRITE(6,6060) M
      ALLOCATE( A(LNA,M), STAT=KERR )
      IF( KERR .NE. 0 ) GOTO 160
      ALLOCATE( JA(LNA,M), STAT=KERR )
      IF( KERR .NE. 0 ) GOTO 170
      CALL DXA005&
      (ITYPE,N,VALUES,IPONTR,IRWIND,NNZ,A,LNA,M,JA,IWK,IERR)
      WRITE(6,6070) IERR
      IF( IERR .EQ. 2000 ) THEN
         WRITE(6,6080) M
         DEALLOCATE(JA)
         DEALLOCATE(A)
         ALLOCATE( A(LNA,M), STAT=KERR )
         IF( KERR .NE. 0) GOTO 160
         ALLOCATE( JA(LNA,M), STAT=KERR )
         IF( KERR .NE. 0) GOTO 170
         CALL DXA005&
         (ITYPE,N,VALUES,IPONTR,IRWIND,NNZ,A,LNA,M,JA,IWK,IERR)
         WRITE(6,6090) IERR
      ENDIF
!
      IF( IERR .GE. 2000 ) GOTO 9999
      WRITE(6,6100)
      DO 180 I=1,N
         WRITE(6,6110) (A(I,J),J=1,M)
  180 CONTINUE
      WRITE(6,6120)
      DO 190 I=1,N
         WRITE(6,6130) (JA(I,J),J=1,M)
  190 CONTINUE
!
 9999 CONTINUE
      DEALLOCATE(JA)
  170 CONTINUE
      DEALLOCATE(A)
  160 CONTINUE
!
      STOP
 6000 FORMAT(/,&
             1X,'***  DXA005  ***',/,/,&
             1X,' **  INPUT  **',/,/,&
             1X,'     N   = ',I5,/,&
             1X,'     NNZ = ',I5,/,&
             1X,'     LNA = ',I5)
 6010 FORMAT(/,&
             1X,'     VECTOR VALUES')
 6020 FORMAT(1X,'       ',F5.1)
 6030 FORMAT(/,&
             1X,'     VECTOR IPONTR')
 6040 FORMAT(1X,'       ',I5)
 6050 FORMAT(/,&
             1X,'     VECTOR IRWIND')
 6060 FORMAT(/,&
             1X,'     INPUT  M   = ',I5,/)
 6070 FORMAT(/,&
             1X,' **  OUTPUT  **',/,/,&
             1X,'     IERR( FIRST  CALL ) = ',I5,/)
 6080 FORMAT(1X,'     OUTPUT M   = ',I5,/)
 6090 FORMAT(1X,'     IERR( SECOND CALL ) = ',I5,/)
 6100 FORMAT(1X,'     MATRIX A')
 6110 FORMAT(1X,'    ',4(2X,F5.1))
 6120 FORMAT(/,&
             1X,'     MATRIX JA')
 6130 FORMAT(1X,'    ',4(2X,I5))
!
      END
```

(d) Output results

```
***  DXA005  ***

 **  INPUT  **

     N   =     5
     NNZ =    16
     LNA =     5

     VECTOR VALUES
         1.0
        -2.1
         4.5
         5.0
         1.1
         2.0
         2.1
         3.0
```

```
       -5.4
        1.2
        4.0
       -5.1
        1.4
        3.5
        4.1
        5.0

    VECTOR IPONTR
          1
          5
          7
         10
         13
         17

    VECTOR IRWIND
          1
          2
          4
          5
          1
          2
          2
          3
          5
          1
          4
          5
          1
          3
          4
          5

    INPUT  M   =     1

**  OUTPUT  **

    IERR( FIRST  CALL ) =  2000

    OUTPUT M   =     4

    IERR( SECOND CALL ) =     0

    MATRIX A
       1.0    1.1    1.2    1.4
       2.0   -2.1    2.1    0.0
       3.0    3.5    0.0    0.0
       4.0    4.5    4.1    0.0
       5.0    5.0   -5.4   -5.1

    MATRIX JA
         1      2      4      5
         2      1      3      0
         3      5      0      0
         4      1      5      0
         5      1      3      4
```

Chapter 3

BASIC MATRIX ALGEBRA

3.1 INTRODUCTION

This chapter describes subroutines that perform basic matrix calculations.

3.1.1 Algorithms Used

3.1.1.1 Real matrix multiplication (speed priority version)

Let A be an $M \times N$ real matrix, let B be an $N \times L$ real matrix, and let the (i,j)-th element of those matrices be a_{ij} and b_{ij}, respectively. If $C = A \times B$ is calculated according to the definition of matrix multiplication as follows:

$$c_{ik} = \sum_{j=1}^{N} a_{ij} \cdot b_{jk}$$

$M \times L \times (2 \times N - 1)$ floating point calculations are required. If the Strassen algorithm, which is described below is used, this number can be reduced by up to 12%. Consider the case when M, N and L are all even numbers. First, subdivide the matrices A, B and C into the submatrices shown below by bisecting each matrix in the row and column directions.

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix}$$

If we define M_1 through M_7 as follows:

$$\begin{aligned} M_1 &= (A_{12} - A_{22}) \cdot (B_{21} + B_{22}) \\ M_2 &= (A_{11} + A_{22}) \cdot (B_{11} + B_{22}) \\ M_3 &= (A_{11} - A_{21}) \cdot (B_{11} + B_{12}) \\ M_4 &= (A_{11} + A_{12}) \cdot B_{22} \\ M_5 &= A_{11} \cdot (B_{12} - B_{22}) \\ M_6 &= A_{22} \cdot (B_{21} - B_{11}) \\ M_7 &= (A_{21} + A_{22}) \cdot B_{11} \end{aligned}$$

the submatrices of C are calculated as follows.

$$\begin{aligned} C_{11} &= M_1 + M_2 - M_4 + M_6 \\ C_{12} &= M_4 + M_5 \\ C_{21} &= M_6 + M_7 \\ C_{22} &= M_2 - M_3 + M_5 - M_7 \end{aligned}$$

If any of the numbers M, N and L is an odd number, first calculate the product by using the method described above for the partial submatrix having the highest order contained in the corresponding matrix. Then, add a correction by calculating the contribution from the elements not contained in that partial submatrix. Also, by using the procedure described above to obtain the product of each submatrix, the number of calculations can further be reduced. This library performs this procedure for up to three steps.

3.2 BASIC CALCULATIONS

3.2.1 DAM1AD, RAM1AD
Adding Real Matrices (Two-Dimensional Array Type)

(1) **Function**

Obtain the sum of two real matrices A and B (two-dimensional array type).

(2) **Usage**

Double precision:

CALL DAM1AD (A, LMA, NM, NN, B, LMB, C, LMC, IERR)

Single precision:

CALL RAM1AD (A, LMA, NM, NN, B, LMB, C, LMC, IERR)

(3) **Arguments**

D:Double precision real Z:Double precision complex
R:Single precision real C:Single precision complex

$$\text{I:}\left\{\begin{array}{l}\text{INTEGER(4) as for 32bit Integer}\\ \text{INTEGER(8) as for 64bit Integer}\end{array}\right\}$$

No.	Argument	Type	Size	Input/ Output	Contents
1	A	$\begin{Bmatrix}\text{D}\\ \text{R}\end{Bmatrix}$	LMA, NN	Input	Real matrix A (two-dimensional array type).
2	LMA	I	1	Input	Adjustable dimension of array A.
3	NM	I	1	Input	Number of rows in matrices A, B and C.
4	NN	I	1	Input	Number of columns in matrices A, B and C.
5	B	$\begin{Bmatrix}\text{D}\\ \text{R}\end{Bmatrix}$	LMB, NN	Input	Real matrix B (two-dimensional array type).
6	LMB	I	1	Input	Adjustable dimension of array B.
7	C	$\begin{Bmatrix}\text{D}\\ \text{R}\end{Bmatrix}$	LMC, NN	Output	Sum of matrices A and B (two-dimensional array type).
8	LMC	I	1	Input	Adjustable dimension of array C.
9	IERR	I	1	Output	Error indicator

(4) **Restrictions**

(a) $\text{NN} > 0$

(b) $0 < \text{NM} \leq \text{LMA}, \text{LMB}, \text{LMC}$

(5) **Error indicator**

IERR value	Meaning	Processing
0	Normal termination.	
1000	NN was equal to 1.	Processing continues.
3000	Restriction (a) or (b) was not satisfied.	Processing is aborted.

(6) **Notes**

None

(7) **Example**

(a) Problem

$$A = \begin{bmatrix} 1 & 2 & 0 & -1 \\ -3 & -5 & 1 & 2 \\ 1 & 3 & 2 & -2 \\ 0 & 2 & 1 & -1 \end{bmatrix}$$

$$B = \begin{bmatrix} -3 & -1 & 1 & -1 \\ -3 & -1 & 0 & 1 \\ -4 & -1 & 1 & 0 \\ -10 & -3 & 1 & 1 \end{bmatrix}$$

Obtain $C = A + B$.

(b) Input data

Matrices A and B, LMA = 11, LMB = 11, LMC = 11, NM = 4 and NN = 4.

(c) Main program

```
      PROGRAM BAM1AD
! *** EXAMPLE OF DAM1AD ***
      IMPLICIT NONE
      INTEGER LMA,LMB,LMC,NM,NN
      PARAMETER( LMA=11, LMB=11, LMC=11 )
      PARAMETER( NM=4, NN=4 )
      INTEGER IERR,I,J
      REAL(8) A(LMA,NN),B(LMB,NN),C(LMC,NN)
!
      DO 100 I=1,NM
         READ(5,*) (A(I,J),J=1,NN)
  100 CONTINUE
      DO 110 I=1,NN
         READ(5,*) (B(I,J),J=1,NN)
  110 CONTINUE
!
      WRITE(6,6000) LMA,LMB,LMC,NM,NN
      DO 120 I=1,NM
         WRITE(6,6010) (A(I,J),J=1,NN)
  120 CONTINUE
      WRITE(6,6020)
      DO 130 I=1,NN
         WRITE(6,6010) (B(I,J),J=1,NN)
  130 CONTINUE
!
      CALL DAM1AD(A,LMA,NM,NN,B,LMB,C,LMC,IERR)
!
      WRITE(6,6030) IERR
      IF( IERR .GE. 3000 ) STOP
!
      WRITE(6,6040)
      DO 140 I=1,NM
         WRITE(6,6010) (C(I,J),J=1,NN)
  140 CONTINUE
      STOP
!
 6000 FORMAT(/,&
             1X,'***  DAM1AD  ***',/,/,&
             1X,' **  INPUT  **',/,/,&
             1X,'     LMA=',I2,'   LMB=',I2,'   LMC=',I2,/,/,&
             1X,'     NM =',I2,'   NN =',I2,/,/,&
             1X,'     INPUT MATRIX A',/)
 6010 FORMAT(1X,6X,11(F7.1))
 6020 FORMAT(/,&
             1X,'     INPUT MATRIX B',/)
 6030 FORMAT(/,&
             1X,' **  OUTPUT  **',/,/,&
             1X,'     IERR = ',I4,/)
 6040 FORMAT(1X,'     OUTPUT MATRIX C',/)
      END
```

(d) Output results

```
***  DAM1AD  ***

 **  INPUT  **
```

```
    LMA=11   LMB=11   LMC=11

    NM = 4  NN = 4

    INPUT MATRIX A

         1.0    2.0    0.0   -1.0
        -3.0   -5.0    1.0    2.0
         1.0    3.0    2.0   -2.0
         0.0    2.0    1.0   -1.0

    INPUT MATRIX B

        -3.0   -1.0    1.0   -1.0
        -3.0   -1.0    0.0    1.0
        -4.0   -1.0    1.0    0.0
       -10.0   -3.0    1.0    1.0

**  OUTPUT  **

    IERR =    0

    OUTPUT MATRIX C

        -2.0    1.0    1.0   -2.0
        -6.0   -6.0    1.0    3.0
        -3.0    2.0    3.0   -2.0
       -10.0   -1.0    2.0    0.0
```

3.2.2 DAM1SB, RAM1SB Subtracting Real Matrices (Two-Dimensional Array Type)

(1) **Function**

Obtain the difference of two real matrices A and B (two-dimensional array type).

(2) **Usage**

Double precision:
  CALL DAM1SB (A, LMA, NM, NN, B, LMB, C, LMC, IERR)
Single precision:
  CALL RAM1SB (A, LMA, NM, NN, B, LMB, C, LMC, IERR)

(3) **Arguments**

D:Double precision real Z:Double precision complex I: { INTEGER(4) as for 32bit Integer, INTEGER(8) as for 64bit Integer }
R:Single precision real C:Single precision complex

No.	Argument	Type	Size	Input/ Output	Contents
1	A	{D R}	LMA, NN	Input	Real matrix A (two-dimensional array type).
2	LMA	I	1	Input	Adjustable dimension of array A.
3	NM	I	1	Input	Number of rows in matrices A, B and C.
4	NN	I	1	Input	Number of columns in matrices A, B and C.
5	B	{D R}	LMB, NN	Input	Real matrix B (two-dimensional array type).
6	LMB	I	1	Input	Adjustable dimension of array B.
7	C	{D R}	LMC, NN	Output	Difference $(A - B)$ of matrices A and B (two-dimensional array type).
8	LMC	I	1	Input	Adjustable dimension of array C.
9	IERR	I	1	Output	Error indicator

(4) **Restrictions**

(a) $\mathrm{NN} > 0$

(b) $0 < \mathrm{NM} \leq \mathrm{LMA}, \mathrm{LMB}, \mathrm{LMC}$

(5) **Error indicator**

IERR value	Meaning	Processing
0	Normal termination.	
1000	NN was equal to 1.	Processing continues.
3000	Restriction (a) or (b) was not satisfied.	Processing is aborted.

(6) **Notes**

None

(7) **Example**

(a) Problem

$$A = \begin{bmatrix} 1 & 2 & 0 & -1 \\ -3 & -5 & 1 & 2 \\ 1 & 3 & 2 & -2 \\ 0 & 2 & 1 & -1 \end{bmatrix}$$

$$B = \begin{bmatrix} -3 & -1 & 1 & -1 \\ -3 & -1 & 0 & 1 \\ -4 & -1 & 1 & 0 \\ -10 & -3 & 1 & 1 \end{bmatrix}$$

Obtain $C = A - B$.

(b) Input data

Matrices A and B, LMA = 11, LMB = 11, LMC = 11, NM = 4 and NN = 4.

(c) Main program

```
      PROGRAM BAM1SB
! *** EXAMPLE OF DAM1SB ***
      IMPLICIT NONE
      INTEGER LMA,LMB,LMC,NM,NN
      PARAMETER( LMA=11, LMB=11, LMC=11 )
      PARAMETER( NM=4, NN=4 )
      INTEGER IERR,I,J
      REAL(8) A(LMA,NN),B(LMB,NN),C(LMC,NN)
!
      DO 100 I=1,NM
         READ(5,*) (A(I,J),J=1,NN)
  100 CONTINUE
      DO 110 I=1,NN
         READ(5,*) (B(I,J),J=1,NN)
  110 CONTINUE
!
      WRITE(6,6000) LMA,LMB,LMC,NM,NN
      DO 120 I=1,NM
         WRITE(6,6010) (A(I,J),J=1,NN)
  120 CONTINUE
      WRITE(6,6020)
      DO 130 I=1,NN
         WRITE(6,6010) (B(I,J),J=1,NN)
  130 CONTINUE
!
      CALL DAM1SB(A,LMA,NM,NN,B,LMB,C,LMC,IERR)
!
      WRITE(6,6030) IERR
      IF( IERR .GE. 3000 ) STOP
!
      WRITE(6,6040)
      DO 140 I=1,NM
         WRITE(6,6010) (C(I,J),J=1,NN)
  140 CONTINUE
      STOP
!
 6000 FORMAT(/,&
             1X,'***  DAM1SB  ***',/,/,&
             1X,' **  INPUT  **',/,/,&
             1X,'     LMA=',I2,'   LMB=',I2,'   LMC=',I2,/,/,&
             1X,'     NM =',I2,'    NN =',I2,/,/,&
             1X,'     INPUT MATRIX A',/)
 6010 FORMAT(1X,6X,11(F7.1))
 6020 FORMAT(/,&
             1X,'     INPUT MATRIX B',/)
 6030 FORMAT(/,&
             1X,' **  OUTPUT  **',/,/,&
             1X,'     IERR = ',I4,/)
 6040 FORMAT(1X,'     OUTPUT MATRIX C',/)
      END
```

(d) Output results

```
***  DAM1SB  ***

 **  INPUT  **
```

```
    LMA=11   LMB=11   LMC=11

    NM = 4   NN = 4

    INPUT MATRIX A

         1.0    2.0    0.0   -1.0
        -3.0   -5.0    1.0    2.0
         1.0    3.0    2.0   -2.0
         0.0    2.0    1.0   -1.0

    INPUT MATRIX B

        -3.0   -1.0    1.0   -1.0
        -3.0   -1.0    0.0    1.0
        -4.0   -1.0    1.0    0.0
       -10.0   -3.0    1.0    1.0

**  OUTPUT  **

    IERR =    0

    OUTPUT MATRIX C

         4.0    3.0   -1.0    0.0
         0.0   -4.0    1.0    1.0
         5.0    4.0    1.0   -2.0
        10.0    5.0    0.0   -2.0
```

3.2.3 DAM1MU, RAM1MU
Multiplying Real Matrices (Two-Dimensional Array Type)

(1) **Function**

Obtain the product of two real matrices A and B (two-dimensional array type).

(2) **Usage**

Double precision:

```
  CALL DAM1MU  (A, LMA, NM, NN, B, LNB, NL, C, LMC, IERR)
```

Single precision:

```
  CALL RAM1MU  (A, LMA, NM, NN, B, LNB, NL, C, LMC, IERR)
```

(3) **Arguments**

D:Double precision real   Z:Double precision complex   I: $\left\{\begin{matrix}\text{INTEGER(4) as for 32bit Integer}\\\text{INTEGER(8) as for 64bit Integer}\end{matrix}\right\}$
R:Single precision real   C:Single precision complex

No.	Argument	Type	Size	Input/ Output	Contents
1	A	$\left\{\begin{matrix}\text{D}\\\text{R}\end{matrix}\right\}$	LMA, NN	Input	Real matrix A (two-dimensional array type).
2	LMA	I	1	Input	Adjustable dimension of array A.
3	NM	I	1	Input	Number of rows in matrix A (Number of rows in matrix C).
4	NN	I	1	Input	Number of columns in matrix A (Number of rows in matrix B).
5	B	$\left\{\begin{matrix}\text{D}\\\text{R}\end{matrix}\right\}$	LNB, NL	Input	Real matrix B (two-dimensional array type).
6	LNB	I	1	Input	Adjustable dimension of array B.
7	NL	I	1	Input	Number of columns in matrix B (Number of columns in matrix C).
8	C	$\left\{\begin{matrix}\text{D}\\\text{R}\end{matrix}\right\}$	LMC, NL	Output	Product ($A \cdot B$) of matrices A and B (two-dimensional array type).
9	LMC	I	1	Input	Adjustable dimension of array C.
10	IERR	I	1	Output	Error indicator

(4) **Restrictions**

(a) $0 < \text{NM} \leq \text{LMA}, \text{LMC}$

(b) $0 < \text{NN} \leq \text{LNB}$

(c) $\text{NL} > 0$

(5) **Error indicator**

IERR value	Meaning	Processing
0	Normal termination.	
1000	NN was equal to 1.	Processing continues.
3000	Restriction (a), (b) or (c) was not satisfied.	Processing is aborted.

(6) **Notes**

Use the speed priority version 3.2.4 $\left\{\begin{matrix}\text{DAM1MS}\\\text{RAM1MS}\end{matrix}\right\}$ when the number is great.

(7) **Example**

(a) Problem

$$A = \begin{bmatrix} 1 & 2 & 0 & -1 \\ -3 & -5 & 1 & 2 \\ 1 & 3 & 2 & -2 \\ 0 & 2 & 1 & -1 \end{bmatrix}$$

$$B = \begin{bmatrix} -3 & -1 & 1 & -1 \\ -3 & -1 & 0 & 1 \\ -4 & -1 & 1 & 0 \\ -10 & -3 & 1 & 1 \end{bmatrix}$$

Obtain $C = AB$.

(b) Input data

Matrices A and B, LMA = 11, LNB = 11, LMC = 11, NM = 4, NN = 4 and NL = 4.

(c) Main program

```
      PROGRAM BAM1MU
! *** EXAMPLE OF DAM1MU ***
      IMPLICIT NONE
      INTEGER LMA,LNB,LMC,NM,NN,NL
      PARAMETER( LMA=11, LNB=11, LMC=11 )
      PARAMETER( NM=4, NN=4, NL=4 )
      INTEGER IERR,I,J
      REAL(8) A(LMA,NN),B(LNB,NL),C(LMC,NL)
!
      DO 100 I=1,NM
         READ(5,*) (A(I,J),J=1,NN)
  100 CONTINUE
      DO 110 I=1,NN
         READ(5,*) (B(I,J),J=1,NL)
  110 CONTINUE
!
      WRITE(6,6000) LMA,LNB,LMC,NM,NN,NL
      DO 120 I=1,NM
         WRITE(6,6010) (A(I,J),J=1,NN)
  120 CONTINUE
      WRITE(6,6020)
      DO 130 I=1,NN
         WRITE(6,6010) (B(I,J),J=1,NL)
  130 CONTINUE
!
      CALL DAM1MU(A,LMA,NM,NN,B,LNB,NL,C,LMC,IERR)
!
      WRITE(6,6030) IERR
      IF( IERR .GE. 3000 ) STOP
!
      WRITE(6,6040)
      DO 140 I=1,NM
         WRITE(6,6010) (C(I,J),J=1,NL)
  140 CONTINUE
      STOP
!
 6000 FORMAT(/,&
             1X,'***  DAM1MU  ***',/,/,&
             1X,' **  INPUT  **',/,/,&
             1X,'     LMA=',I2,'   LNB=',I2,'   LMC=',I2,/,/,&
             1X,'     NM =',I2,'   NN =',I2,'   NL =',I2,/,/,&
             1X,'     INPUT MATRIX A',/)
 6010 FORMAT(1X,6X,11(F7.1))
 6020 FORMAT(/,&
             1X,’     INPUT MATRIX B’,/)
 6030 FORMAT(/,&
             1X,’ **  OUTPUT  **’,/,/,&
             1X,’     IERR = ’,I4,/)
 6040 FORMAT(1X,’     OUTPUT MATRIX C’,/)
      END
```

(d) Output results

```
***  DAM1MU  ***

 **  INPUT  **

     LMA=11   LNB=11   LMC=11

     NM = 4   NN = 4   NL = 4

     INPUT MATRIX A

          1.0    2.0    0.0   -1.0
         -3.0   -5.0    1.0    2.0
          1.0    3.0    2.0   -2.0
          0.0    2.0    1.0   -1.0

     INPUT MATRIX B

         -3.0   -1.0    1.0   -1.0
         -3.0   -1.0    0.0    1.0
         -4.0   -1.0    1.0    0.0
        -10.0   -3.0    1.0    1.0

 **  OUTPUT  **

     IERR =    0

     OUTPUT MATRIX C

          1.0    0.0    0.0    0.0
          0.0    1.0    0.0    0.0
          0.0    0.0    1.0    0.0
          0.0    0.0    0.0    1.0
```

3.2.4 DAM1MS, RAM1MS Multiplying Real Matrices (Two-Dimensional Array Type) (Speed Priority Version)

(1) **Function**

Use the Strassen algorithm to obtain the product of two real matrices A and B (two-dimensional array type).

(2) **Usage**

Double precision:
CALL DAM1MS (A, LMA, M, N, B, LNB, K, C, LMC, W1, IERR)
Single precision:
CALL RAM1MS (A, LMA, M, N, B, LNB, K, C, LMC, W1, IERR)

(3) **Arguments**

D:Double precision real Z:Double precision complex
R:Single precision real C:Single precision complex
I: $\left\{\begin{array}{l}\text{INTEGER(4) as for 32bit Integer}\\ \text{INTEGER(8) as for 64bit Integer}\end{array}\right\}$

No.	Argument	Type	Size	Input/ Output	Contents
1	A	$\left\{\begin{array}{c}\text{D}\\ \text{R}\end{array}\right\}$	LMA, N	Input	Real matrix A (two-dimensional array type).
2	LMA	I	1	Input	Adjustable dimension of array A.
3	M	I	1	Input	Number of rows in matrix A (Number of rows in matrix C).
4	N	I	1	Input	Number of columns in matrix A (Number of rows in matrix B).
5	B	$\left\{\begin{array}{c}\text{D}\\ \text{R}\end{array}\right\}$	LNB, K	Input	Real matrix B (two-dimensional array type).
6	LNB	I	1	Input	Adjustable dimension of array B.
7	K	I	1	Input	Number of columns in matrix B (Number of columns in matrix C).
8	C	$\left\{\begin{array}{c}\text{D}\\ \text{R}\end{array}\right\}$	LMC, K	Output	Real matrix C (two-dimensional array type).
9	LMC	I	1	Input	Adjustable dimension of array C.
10	W1	$\left\{\begin{array}{c}\text{D}\\ \text{R}\end{array}\right\}$	See Contents	Work	Work area **Size**: $(\mathrm{M}\times\mathrm{N}+\mathrm{N}\times\mathrm{K}+\mathrm{K}\times\mathrm{M})/3$
11	IERR	I	1	Output	Error indicator

(4) **Restrictions**

(a) $0 < \mathrm{M} \le \mathrm{LMA}, \mathrm{LMC}$

(b) $0 < \mathrm{N} \le \mathrm{LNB}$

(c) $\mathrm{K} > 0$

(5) **Error indicator**

IERR value	Meaning	Processing
0	Normal termination.	
1000	N was equal to 1.	Processing continues.
3000	Restriction (a), (b) or (c) was not satisfied.	Processing is aborted.

(6) **Notes**

(a) Since the 3.2.4 $\left\{\begin{array}{c}\text{DAM1MS}\\ \text{RAM1MS}\end{array}\right\}$ use memory than the 3.2.3 $\left\{\begin{array}{c}\text{DAM1MU}\\ \text{RAM1MU}\end{array}\right\}$ because of the portion used for the work area, if sufficient memory cannot be allocated, the 3.2.3 $\left\{\begin{array}{c}\text{DAM1MU}\\ \text{RAM1MU}\end{array}\right\}$ should be used.

(b) The result obtained by using the 3.2.4 $\left\{\begin{array}{c}\text{DAM1MS}\\ \text{RAM1MS}\end{array}\right\}$ may be somewhat less precise than the result obtained by using 3.2.3 $\left\{\begin{array}{c}\text{DAM1MU}\\ \text{RAM1MU}\end{array}\right\}$.

(7) **Example**

(a) Problem

$$A = \begin{bmatrix} 1 & 2 & 0 & -1 \\ -3 & -5 & 1 & 2 \\ 1 & 3 & 2 & -2 \\ 0 & 2 & 1 & -1 \end{bmatrix}$$

$$B = \begin{bmatrix} -3 & -1 & 1 & -1 \\ -3 & -1 & 0 & 1 \\ -4 & -1 & 1 & 0 \\ -10 & -3 & 1 & 1 \end{bmatrix}$$

Obtain $C = AB$.

(b) Input data

Matrices A and B, LMA = 11, LNB = 11, LMC = 11, M = 4, N = 4 and K = 4.

(c) Main program

```
      PROGRAM BAM1MS
! *** EXAMPLE OF DAM1MS ***
      IMPLICIT NONE
      INTEGER LMA,LNB,LMC,M,N,K,IWK
      PARAMETER( LMA=11, LNB=11, LMC=11 )
      PARAMETER( M=4, N=4, K=4 )
      PARAMETER( IWK = (M*N+N*K+K*M)/3 )
      INTEGER IERR,I,J
      REAL(8) A(LMA,N),B(LNB,K),C(LMC,K),W1(IWK)
!
      DO 100 I=1,M
         READ(5,*) (A(I,J),J=1,N)
  100 CONTINUE
      DO 110 I=1,N
         READ(5,*) (B(I,J),J=1,K)
  110 CONTINUE
!
      WRITE(6,6000) LMA,LNB,LMC,M,N,K
      DO 120 I=1,M
         WRITE(6,6010) (A(I,J),J=1,N)
  120 CONTINUE
      WRITE(6,6020)
      DO 130 I=1,N
         WRITE(6,6010) (B(I,J),J=1,K)
  130 CONTINUE
```

```
!
      CALL DAM1MS(A,LMA,M,N,B,LNB,K,C,LMC,W1,IERR)
!
      WRITE(6,6030) IERR
      IF( IERR .GE. 3000 ) STOP
!
      WRITE(6,6040)
      DO 140 I=1,M
         WRITE(6,6010) (C(I,J),J=1,K)
  140 CONTINUE
      STOP
!
 6000 FORMAT(/,&
             1X,'***  DAM1MS  ***',/,/,&
             1X,' **  INPUT  **',/,/,&
             1X,'     LMA=',I2,'   LNB=',I2,'   LMC=',I2,/,/,&
             1X,'     M  =',I2,'   N  =',I2,'   K  =',I2,/,/,&
             1X,'     INPUT MATRIX A',/)
 6010 FORMAT(1X,6X,11(F7.1))
 6020 FORMAT(/,&
             1X,'     INPUT MATRIX B',/)
 6030 FORMAT(/,&
             1X,' **  OUTPUT  **',/,/,&
             1X,'     IERR = ',I4,/)
 6040 FORMAT(1X,'     OUTPUT MATRIX C',/)
      END
```

(d) Output results

```
***  DAM1MS  ***

 **  INPUT  **

     LMA=11   LNB=11   LMC=11

     M  = 4   N  = 4   K  = 4

     INPUT MATRIX A

          1.0    2.0    0.0   -1.0
         -3.0   -5.0    1.0    2.0
          1.0    3.0    2.0   -2.0
          0.0    2.0    1.0   -1.0

     INPUT MATRIX B

         -3.0   -1.0    1.0   -1.0
         -3.0   -1.0    0.0    1.0
         -4.0   -1.0    1.0    0.0
        -10.0   -3.0    1.0    1.0

 **  OUTPUT  **

     IERR =    0

     OUTPUT MATRIX C

          1.0    0.0    0.0    0.0
          0.0    1.0    0.0    0.0
          0.0    0.0    1.0    0.0
          0.0    0.0    0.0    1.0
```

3.2.5 DAMT1M, RAMT1M Multiplying a Real Matrix (Two-Dimensional Array Type) and Its Transpose Matrix

(1) **Function**

Obtain the product of the real matrix A (two-dimensional array type) and its transpose matrix.

(2) **Usage**

Double precision:

CALL DAMT1M (A, LMA, NM, NN, B, LMB, IERR)

Single precision:

CALL RAMT1M (A, LMA, NM, NN, B, LMB, IERR)

(3) **Arguments**

D:Double precision real Z:Double precision complex
R:Single precision real C:Single precision complex
I: $\left\{\begin{matrix}\text{INTEGER(4) as for 32bit Integer}\\ \text{INTEGER(8) as for 64bit Integer}\end{matrix}\right\}$

No.	Argument	Type	Size	Input/ Output	Contents
1	A	$\left\{\begin{matrix}\text{D}\\ \text{R}\end{matrix}\right\}$	LMA, NN	Input	Real matrix A (two-dimensional array type).
2	LMA	I	1	Input	Adjustable dimension of array A.
3	NM	I	1	Input	Number of rows in matrix A.
4	NN	I	1	Input	Number of columns in matrix A.
5	B	$\left\{\begin{matrix}\text{D}\\ \text{R}\end{matrix}\right\}$	LMB, NM	Output	Product $(A \cdot A^T)$ (two-dimensional array type) of matrix A and its transpose matrix.
6	LMB	I	1	Input	Adjustable dimension of array B.
7	IERR	I	1	Output	Error indicator

(4) **Restrictions**

(a) $\mathrm{NN} > 0$, $\mathrm{NM} > 0$

(b) $\mathrm{NM} \leq \mathrm{LMA}, \mathrm{LMB}$

(5) **Error indicator**

IERR value	Meaning	Processing
0	Normal termination.	
1000	NN was equal to 1.	Processing continues.
3000	Restriction (a) or (b) was not satisfied.	Processing is aborted.

(6) **Notes**

None

(7) **Example**

(a) Problem

$$A = \begin{bmatrix} 1 & 2 & 3 & 4 \\ -2 & 3 & 4 & 5 \\ -3 & -4 & 5 & 6 \\ -4 & -5 & -6 & 7 \end{bmatrix}$$

Obtain $B = AA^T$.

(b) Input data

Matrix A, LMA = 11, LMB = 11, NM = 4 and NN = 4.

(c) Main program

```
      PROGRAM BAMT1M
! *** EXAMPLE OF DAMT1M ***
      IMPLICIT NONE
      INTEGER LMA,LMB,NM,NN
      PARAMETER( LMA=11, LMB=11 )
      PARAMETER( NM=4, NN=4 )
      INTEGER IERR,I,J
      REAL(8) A(LMA,NN),B(LMB,NM)
!
      DO 100 I=1,NM
         READ(5,*) (A(I,J),J=1,NN)
  100 CONTINUE
!
      WRITE(6,6000) LMA,LMB,NM,NN
      DO 110 I=1,NM
         WRITE(6,6010) (A(I,J),J=1,NN)
  110 CONTINUE
!
      CALL DAMT1M(A,LMA,NM,NN,B,LMB,IERR)
!
      WRITE(6,6020) IERR
      IF( IERR .GE. 3000 ) STOP
!
      WRITE(6,6030)
      DO 120 I=1,NM
         WRITE(6,6010) (B(I,J),J=1,NM)
  120 CONTINUE
      STOP
!
 6000 FORMAT(/,&
             1X,'***  DAMT1M  ***',/,/,&
             1X,' **  INPUT  **',/,/,&
             1X,'     LMA=',I2,'   LMB=',I2,'   NM=',I2,'   NN=',I2,/,/,&
             1X,'     INPUT MATRIX A',/)
 6010 FORMAT(1X,6X,11(F7.1))
 6020 FORMAT(/,&
             1X,' **  OUTPUT  **',/,/,&
             1X,'     IERR = ',I4,/)
 6030 FORMAT(1X,'     OUTPUT MATRIX B',/)
      END
```

(d) Output results

```
***  DAMT1M  ***

 **  INPUT  **

     LMA=11   LMB=11   NM= 4   NN= 4

     INPUT MATRIX A

         1.0    2.0    3.0    4.0
        -2.0    3.0    4.0    5.0
        -3.0   -4.0    5.0    6.0
        -4.0   -5.0   -6.0    7.0

 **  OUTPUT  **

     IERR =    0

     OUTPUT MATRIX B

        30.0   36.0   28.0   -4.0
        36.0   54.0   44.0    4.0
        28.0   44.0   86.0   44.0
        -4.0    4.0   44.0  126.0
```

3.2.6 DATM1M, RATM1M Multiplying the Transpose Matrix of a Real Matrix (Two-Dimensional Array Type) and the Original Matrix

(1) **Function**

Obtain the product of the transpose matrix of the real matrix A (two-dimensional array type) and the original matrix.

(2) **Usage**

Double precision:
CALL DATM1M (A, LMA, NM, NN, B, LNB, IERR)
Single precision:
CALL RATM1M (A, LMA, NM, NN, B, LNB, IERR)

(3) **Arguments**

D:Double precision real Z:Double precision complex I: $\left\{\begin{array}{l}\text{INTEGER(4) as for 32bit Integer}\\\text{INTEGER(8) as for 64bit Integer}\end{array}\right\}$
R:Single precision real C:Single precision complex

No.	Argument	Type	Size	Input/ Output	Contents
1	A	$\left\{\begin{matrix}D\\R\end{matrix}\right\}$	LMA, NN	Input	Real matrix A (two-dimensional array type).
2	LMA	I	1	Input	Adjustable dimension of array A.
3	NM	I	1	Input	Number of rows in matrix A.
4	NN	I	1	Input	Number of columns in matrix A.
5	B	$\left\{\begin{matrix}D\\R\end{matrix}\right\}$	LNB, NN	Output	Product ($A^T \cdot A$) of the transpose matrix of matrix A and the original matrix.
6	LNB	I	1	Input	Adjustable dimension of array B.
7	IERR	I	1	Output	Error indicator

(4) **Restrictions**

(a) $0 < \text{NN} \leq \text{LNB}$

(b) $0 < \text{NM} \leq \text{LMA}$

(5) **Error indicator**

IERR value	Meaning	Processing
0	Normal termination.	
1000	NN was equal to 1.	Processing continues.
3000	Restriction (a) or (b) was not satisfied.	Processing is aborted.

(6) **Notes**

None

(7) **Example**

(a) Problem

$$A = \begin{bmatrix} 1 & 2 & 3 & 4 \\ -2 & 3 & 4 & 5 \\ -3 & -4 & 5 & 6 \\ -4 & -5 & -6 & 7 \end{bmatrix}$$

$$B = \begin{bmatrix} -3 & -1 & 1 & -1 \\ -3 & -1 & 0 & 1 \\ -4 & -1 & 1 & 0 \\ -10 & -3 & 1 & 1 \end{bmatrix}$$

Obtain $B = A^T A$.

(b) Input data

Matrix A, LMA = 11, LNB = 11, NM = 4 and NN = 4.

(c) Main program

```
      PROGRAM BATM1M
! *** EXAMPLE OF DATM1M ***
      IMPLICIT NONE
      INTEGER LMA,LNB,NM,NN
      PARAMETER( LMA=11, LNB=11 )
      PARAMETER( NM=4, NN=4 )
      INTEGER IERR,I,J
      REAL(8) A(LMA,NN),B(LNB,NN)
!
      DO 100 I=1,NM
         READ(5,*) (A(I,J),J=1,NN)
  100 CONTINUE
!
      WRITE(6,6000) LMA,LNB,NM,NN
      DO 110 I=1,NM
         WRITE(6,6010) (A(I,J),J=1,NN)
  110 CONTINUE
!
      CALL DATM1M(A,LMA,NM,NN,B,LNB,IERR)
!
      WRITE(6,6020) IERR
      IF( IERR .GE. 3000 ) STOP
!
      WRITE(6,6030)
      DO 120 I=1,NN
         WRITE(6,6010) (B(I,J),J=1,NN)
  120 CONTINUE
      STOP
!
 6000 FORMAT(/,&
             1X,'***  DATM1M  ***',/,/,&
             1X,' **  INPUT  **',/,/,&
             1X,'     LMA=',I2,'   LNB=',I2,'   NM=',I2,'   NN=',I2,/,/,&
             1X,'     INPUT MATRIX A',/)
 6010 FORMAT(1X,6X,11(F7.1))
 6020 FORMAT(/,&
             1X,' **  OUTPUT  **',/,/,&
             1X,'     IERR = ',I4,/)
 6030 FORMAT(1X,'     OUTPUT MATRIX B',/)
      END
```

(d) Output results

```
***  DATM1M  ***

 **  INPUT  **

     LMA=11   LNB=11   NM= 4   NN= 4

     INPUT MATRIX A

          1.0    2.0    3.0    4.0
         -2.0    3.0    4.0    5.0
         -3.0   -4.0    5.0    6.0
         -4.0   -5.0   -6.0    7.0

 **  OUTPUT  **

     IERR =    0

     OUTPUT MATRIX B

         30.0   28.0    4.0  -52.0
```

```
    28.0    54.0    28.0   -36.0
     4.0    28.0    86.0    20.0
   -52.0   -36.0    20.0   126.0
```

3.2.7 DAM1MM, RAM1MM Multiplying Real Matrices (Two-Dimensional Array Type) ($C = C \pm AB$)

(1) **Function**

Obtain the product of real matrices A and B ($C = C \pm AB$).

(2) **Usage**

Double precision:

```
CALL DAM1MM  (A, LMA, NM, NN, B, LNB, NL, C, LMC, ISW, IERR)
```

Single precision:

```
CALL RAM1MM  (A, LMA, NM, NN, B, LNB, NL, C, LMC, ISW, IERR)
```

(3) **Arguments**

D:Double precision real   Z:Double precision complex   I: $\left\{\begin{array}{l}\text{INTEGER(4) as for 32bit Integer}\\ \text{INTEGER(8) as for 64bit Integer}\end{array}\right\}$
R:Single precision real   C:Single precision complex

No.	Argument	Type	Size	Input/ Output	Contents
1	A	$\left\{\begin{array}{c}\text{D}\\ \text{R}\end{array}\right\}$	LMA, NN	Input	Real matrix A (two-dimensional array type).
2	LMA	I	1	Input	Adjustable dimension of array A.
3	NM	I	1	Input	Number of rows in matrix A (Number of rows in matrix C).
4	NN	I	1	Input	Number of columns in matrix A (Number of rows in matrix B).
5	B	$\left\{\begin{array}{c}\text{D}\\ \text{R}\end{array}\right\}$	LNB, NL	Input	Real matrix B (two-dimensional array type).
6	LNB	I	1	Input	Adjustable dimension of array B.
7	NL	I	1	Input	Number of columns in matrix B (Number of columns in matrix C).
8	C	$\left\{\begin{array}{c}\text{D}\\ \text{R}\end{array}\right\}$	LMC, NL	Input	Initial real matrix C (when ISW = ±1) (two-dimensional array type).
				Output	Product of real matrices ($C = [C\pm]AB$).
9	LMC	I	1	Input	Adjustable dimension of array C.
10	ISW	I	1	Input	Processing switch. ISW = 1: Obtain $C = C + AB$. ISW = 0: Obtain $C = AB$. ISW = −1: Obtain $C = C - AB$.
11	IERR	I	1	Output	Error indicator

(4) **Restrictions**

(a) $0 < \mathrm{NM} \leq \mathrm{LMA}, \mathrm{LMC}$

(b) $0 < \mathrm{NN} \leq \mathrm{LNB}$

(c) $\mathrm{NL} > 0$

(d) $\mathrm{ISW} \in \{0, 1, -1\}$

(5) **Error indicator**

IERR value	Meaning	Processing
0	Normal termination.	
1000	NN was equal to 1.	Processing continues.
3000	Restriction (a), (b) or (c) was not satisfied.	Processing is aborted.
3010	Restriction (d) was not satisfied.	

(6) **Notes**
None

(7) **Example**

(a) Problem

$$A = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 3 & 3 \\ 4 & 4 & 4 & 4 & 4 \end{bmatrix}$$

$$B = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 3 & 3 & 3 \\ 4 & 4 & 4 & 4 & 4 & 4 \\ 5 & 5 & 5 & 5 & 5 & 5 \end{bmatrix}$$

Obtain $C = AB$.

(b) Input data
Matrices A and B, LMA = 11, LNB = 11, LNC = 11, NM = 4, NN = 5, NL = 6 and ISW = 0.

(c) Main program

```
      PROGRAM BAM1MM
! *** EXAMPLE OF DAM1MM ***
      IMPLICIT NONE
      INTEGER LMA,LNB,LMC,ISW,NM,NN,NL
      PARAMETER( LMA=11, LNB=11, LMC=11, ISW=0 )
      PARAMETER( NM=4, NN=5, NL=6 )
      INTEGER IERR,I,J
      REAL(8) A(LMA,NN),B(LNB,NL),C(LMC,NL)
!
      DO 100 I=1,NM
         READ(5,*) (A(I,J),J=1,NN)
  100 CONTINUE
      DO 110 I=1,NN
         READ(5,*) (B(I,J),J=1,NL)
  110 CONTINUE
!
      WRITE(6,6000) LMA,LNB,LMC,NM,NN,NL,ISW
      WRITE(6,6010) 'MATRIX A'
      DO 120 I=1,NM
         WRITE(6,6020) (A(I,J),J=1,NN)
  120 CONTINUE
      WRITE(6,6010) 'MATRIX B'
      DO 130 I=1,NN
```

```
            WRITE(6,6020) (B(I,J),J=1,NL)
  130 CONTINUE
!
      CALL DAM1MM(A,LMA,NM,NN,B,LNB,NL,C,LMC,ISW,IERR)
!
      WRITE(6,6030) IERR
      IF( IERR .GE. 3000 ) STOP
!
      WRITE(6,6040) 'MATRIX C'
      DO 140 I=1,NM
         WRITE(6,6020) (C(I,J),J=1,NL)
  140 CONTINUE
      STOP
!
 6000 FORMAT(/,&
             1X,'***  DAM1MM  ***',/,/,&
             1X,' **  INPUT  **',/,/,&
             1X,'     LMA=',I2,'   LNB=',I2,'   LMC=',I2,/,/,&
             1X,'     NM =',I2,'   NN =',I2,'   NL =',I2,/,/,&
             1X,'     ISW=',I2,/,/,&
             1X,'     INPUT MATRIX')
 6010 FORMAT(/,&
             1X,5X,A)
 6020 FORMAT(1X,6X,11(F7.1))
 6030 FORMAT(/,&
             1X,' **  OUTPUT  **',/,/,&
             1X,'     IERR = ',I4,/)
 6040 FORMAT(1X,'     OUTPUT MATRIX',/,/,&
             1X,5X,A)
      END
```

(d) Output results

```
***  DAM1MM  ***

 **  INPUT  **

     LMA=11   LNB=11   LMC=11

     NM = 4   NN = 5   NL = 6

     ISW= 0

     INPUT MATRIX

     MATRIX A
         1.0    1.0    1.0    1.0    1.0
         2.0    2.0    2.0    2.0    2.0
         3.0    3.0    3.0    3.0    3.0
         4.0    4.0    4.0    4.0    4.0

     MATRIX B
         1.0    1.0    1.0    1.0    1.0    1.0
         2.0    2.0    2.0    2.0    2.0    2.0
         3.0    3.0    3.0    3.0    3.0    3.0
         4.0    4.0    4.0    4.0    4.0    4.0
         5.0    5.0    5.0    5.0    5.0    5.0

 **  OUTPUT  **

     IERR =    0

     OUTPUT MATRIX

     MATRIX C
        15.0   15.0   15.0   15.0   15.0   15.0
        30.0   30.0   30.0   30.0   30.0   30.0
        45.0   45.0   45.0   45.0   45.0   45.0
        60.0   60.0   60.0   60.0   60.0   60.0
```

3.2.8 DAM1MT, RAM1MT
Multiplying Real Matrices (Two-Dimensional Array Type) $(C = C \pm AB^T)$

(1) **Function**

Obtain the product of real matrices A and B $(C = [C\pm]AB^T)$.

(2) **Usage**

Double precision:

CALL DAM1MT (A, LMA, NM, NN, B, LLB, NL, C, LMC, ISW, IERR)

Single precision:

CALL RAM1MT (A, LMA, NM, NN, B, LLB, NL, C, LMC, ISW, IERR)

(3) **Arguments**

D:Double precision real Z:Double precision complex
R:Single precision real C:Single precision complex
I: {INTEGER(4) as for 32bit Integer / INTEGER(8) as for 64bit Integer}

No.	Argument	Type	Size	Input/ Output	Contents
1	A	{D / R}	LMA, NN	Input	Real matrix A (two-dimensional array type).
2	LMA	I	1	Input	Adjustable dimension of array A.
3	NM	I	1	Input	Number of rows in matrix A (Number of rows in matrix C).
4	NN	I	1	Input	Number of columns in matrix A (Number of columns in matrix B).
5	B	{D / R}	LLB, NN	Input	Real matrix B (two-dimensional array type).
6	LLB	I	1	Input	Adjustable dimension of array B.
7	NL	I	1	Input	Number of rows in matrix B (Number of columns in matrix C).
8	C	{D / R}	LMC, NL	Input	Initial real matrix C (when ISW = ±1) (two-dimensional array type).
				Output	Product of real matrices $(C = [C\pm]AB^T)$.
9	LMC	I	1	Input	Adjustable dimension of array C.
10	ISW	I	1	Input	Processing switch. ISW = 1: Obtain $C = C + AB^T$. ISW = 0: Obtain $C = AB^T$. ISW = −1: Obtain $C = C - AB^T$.
11	IERR	I	1	Output	Error indicator

(4) Restrictions

(a) $0 < \mathrm{NM} \leq \mathrm{LMA}, \mathrm{LMC}$

(b) $0 < \mathrm{NL} \leq \mathrm{LLB}$

(c) $\mathrm{NN} > 0$

(d) $\mathrm{ISW} \in \{0, 1, -1\}$

(5) Error indicator

IERR value	Meaning	Processing
0	Normal termination.	
1000	NN was equal to 1.	Processing continues.
3000	Restriction (a), (b) or (c) was not satisfied.	Processing is aborted.
3010	Restriction (d) was not satisfied.	

(6) Notes

None

(7) Example

(a) Problem

$$A = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 3 & 3 \\ 4 & 4 & 4 & 4 & 4 \end{bmatrix}$$

$$B = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 3 & 3 \\ 4 & 4 & 4 & 4 & 4 \\ 5 & 5 & 5 & 5 & 5 \end{bmatrix}$$

Obtain $C = AB^T$.

(b) Input data

Matrices A and B, LMA = 11, LLB = 11, LNC = 11, NM = 4, NN = 5, NL = 5 and ISW = 0.

(c) Main program

```
      PROGRAM BAM1MT
! *** EXAMPLE OF DAM1MT ***
      IMPLICIT NONE
      INTEGER LMA,LLB,LMC,ISW,NM,NN,NL
      PARAMETER( LMA=11, LLB=11, LMC=11, ISW=0 )
      PARAMETER( NM=4, NN=5, NL=5 )
      INTEGER IERR,I,J
      REAL(8) A(LMA,NN),B(LLB,NN),C(LMC,NL)
!
      DO 100 I=1,NM
         READ(5,*) (A(I,J),J=1,NN)
  100 CONTINUE
      DO 110 I=1,NL
         READ(5,*) (B(I,J),J=1,NN)
  110 CONTINUE
!
      WRITE(6,6000) LMA,LLB,LMC,NM,NN,NL,ISW
      WRITE(6,6010) 'MATRIX A'
      DO 120 I=1,NM
         WRITE(6,6020) (A(I,J),J=1,NN)
  120 CONTINUE
      WRITE(6,6010) 'MATRIX B'
      DO 130 I=1,NL
```

```
           WRITE(6,6020) (B(I,J),J=1,NN)
  130 CONTINUE
!
      CALL DAM1MT(A,LMA,NM,NN,B,LLB,NL,C,LMC,ISW,IERR)
!
      WRITE(6,6030) IERR
      IF( IERR .GE. 3000 ) STOP
!
      WRITE(6,6040) 'MATRIX C'
      DO 140 I=1,NM
         WRITE(6,6020) (C(I,J),J=1,NL)
  140 CONTINUE
      STOP
!
 6000 FORMAT(/,&
             1X,'***  DAM1MT  ***',/,/,&
             1X,' **  INPUT  **',/,/,&
             1X,'     LMA=',I2,'   LLB=',I2,'   LMC=',I2,/,/,&
             1X,'     NM =',I2,'   NN =',I2,'   NL =',I2,/,/,&
             1X,'     ISW=',I2,/,/,&
             1X,'     INPUT MATRIX')
 6010 FORMAT(/,&
             1X,5X,A)
 6020 FORMAT(1X,6X,11(F7.1))
 6030 FORMAT(/,&
             1X,' **  OUTPUT  **',/,/,&
             1X,'     IERR = ',I4,/)
 6040 FORMAT(1X,'     OUTPUT MATRIX',/,/,&
             1X,5X,A)
      END
```

(d) Output results

```
***  DAM1MT  ***

 **  INPUT  **

     LMA=11   LLB=11   LMC=11

     NM = 4   NN = 5   NL = 5

     ISW= 0

     INPUT MATRIX

     MATRIX A
         1.0    1.0    1.0    1.0    1.0
         2.0    2.0    2.0    2.0    2.0
         3.0    3.0    3.0    3.0    3.0
         4.0    4.0    4.0    4.0    4.0

     MATRIX B
         1.0    1.0    1.0    1.0    1.0
         2.0    2.0    2.0    2.0    2.0
         3.0    3.0    3.0    3.0    3.0
         4.0    4.0    4.0    4.0    4.0
         5.0    5.0    5.0    5.0    5.0

 **  OUTPUT  **

     IERR =    0

     OUTPUT MATRIX

     MATRIX C
         5.0   10.0   15.0   20.0   25.0
        10.0   20.0   30.0   40.0   50.0
        15.0   30.0   45.0   60.0   75.0
        20.0   40.0   60.0   80.0  100.0
```

3.2.9 DAM1TM, RAM1TM
Multiplying Real Matrices (Two-Dimensional Array Type) $(C = C \pm A^T B)$

(1) **Function**

Obtain the product of real matrices A and B $(C = [C\pm]A^T B)$.

(2) **Usage**

Double precision:
  CALL DAM1TM (A, LNA, NM, NN, B, LNB, NL, C, LMC, ISW, IERR)
Single precision:
  CALL RAM1TM (A, LNA, NM, NN, B, LNB, NL, C, LMC, ISW, IERR)

(3) **Arguments**

D:Double precision real Z:Double precision complex I: $\left\{\begin{array}{l}\text{INTEGER(4) as for 32bit Integer}\\ \text{INTEGER(8) as for 64bit Integer}\end{array}\right\}$
R:Single precision real C:Single precision complex

No.	Argument	Type	Size	Input/ Output	Contents
1	A	$\left\{\begin{array}{c}D\\R\end{array}\right\}$	LNA, NM	Input	Real matrix A (two-dimensional array type).
2	LNA	I	1	Input	Adjustable dimension of array A.
3	NM	I	1	Input	Number of columns in matrix A (Number of rows in matrix C).
4	NN	I	1	Input	Number of rows in matrix A (Number of rows in matrix B).
5	B	$\left\{\begin{array}{c}D\\R\end{array}\right\}$	LNB, NL	Input	Real matrix B (two-dimensional array type).
6	LNB	I	1	Input	Adjustable dimension of array B.
7	NL	I	1	Input	Number of columns in matrix B (Number of columns in matrix C).
8	C	$\left\{\begin{array}{c}D\\R\end{array}\right\}$	LMC, NL	Input	Initial real matrix C (when ISW = ±1) (two-dimensional array type).
				Output	Product of real matrices $(C = [C\pm]A^T B)$.
9	LMC	I	1	Input	Adjustable dimension of array C.
10	ISW	I	1	Input	Processing switch. ISW = 1: Obtain $C = C + A^T B$. ISW = 0: Obtain $C = A^T B$. ISW = −1: Obtain $C = C - A^T B$.
11	IERR	I	1	Output	Error indicator

(4) **Restrictions**

(a) $0 < \mathrm{NM} \leq \mathrm{LMC}$

(b) $0 < \mathrm{NN} \leq \mathrm{LNA}, \mathrm{LNB}$

(c) $\mathrm{NL} > 0$

(d) $\mathrm{ISW} \in \{0, 1, -1\}$

(5) **Error indicator**

IERR value	Meaning	Processing
0	Normal termination.	
1000	NN was equal to 1.	Processing continues.
3000	Restriction (a), (b) or (c) was not satisfied.	Processing is aborted.
3010	Restriction (d) was not satisfied.	

(6) **Notes**

None

(7) **Example**

(a) Problem

$$A = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 3 & 3 \\ 4 & 4 & 4 & 4 & 4 \\ 5 & 5 & 5 & 5 & 5 \end{bmatrix}$$

$$B = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 3 \\ 4 & 4 & 4 & 4 \\ 5 & 5 & 5 & 5 \end{bmatrix}$$

Obtain $C = A^T B$.

(b) Input data

Matrices A and B, LNA = 11, LNB = 11, LNC = 11, NM = 5, NN = 5, NL = 4 and ISW = 0.

(c) Main program

```
      PROGRAM BAM1TM
! *** EXAMPLE OF DAM1TM ***
      IMPLICIT NONE
      INTEGER LNA,LNB,LMC,ISW,NM,NN,NL
      PARAMETER( LNA=11, LNB=11, LMC=11, ISW=0 )
      PARAMETER( NM=5, NN=5, NL=4 )
      INTEGER IERR,I,J
      REAL(8) A(LNA,NM),B(LNB,NL),C(LMC,NL)
!
      DO 100 I=1,NN
         READ(5,*) (A(I,J),J=1,NM)
  100 CONTINUE
      DO 110 I=1,NN
         READ(5,*) (B(I,J),J=1,NL)
  110 CONTINUE
!
      WRITE(6,6000) LNA,LNB,LMC,NM,NN,NL,ISW
      WRITE(6,6010) 'MATRIX A'
      DO 120 I=1,NN
         WRITE(6,6020) (A(I,J),J=1,NM)
  120 CONTINUE
```

```
      WRITE(6,6010) 'MATRIX B'
      DO 130 I=1,NN
         WRITE(6,6020) (B(I,J),J=1,NL)
  130 CONTINUE
!
      CALL DAM1TM(A,LNA,NM,NN,B,LNB,NL,C,LMC,ISW,IERR)
!
      WRITE(6,6030) IERR
      IF( IERR .GE. 3000 ) STOP
!
      WRITE(6,6040) 'MATRIX C'
      DO 140 I=1,NM
         WRITE(6,6020) (C(I,J),J=1,NL)
  140 CONTINUE
      STOP
!
 6000 FORMAT(/,&
             1X,'***  DAM1TM  ***',/,/,&
             1X,' **  INPUT  **',/,/,&
             1X,'     LNA=',I2,'   LNB=',I2,'   LMC=',I2,/,/,&
             1X,'     NM =',I2,'   NN =',I2,'   NL =',I2,/,/,&
             1X,'     ISW=',I2,/,/,&
             1X,'     INPUT MATRIX')
 6010 FORMAT(/,&
             1X,5X,A)
 6020 FORMAT(1X,6X,11(F7.1))
 6030 FORMAT(/,&
             1X,' **  OUTPUT  **',/,/,&
             1X,'     IERR = ',I4,/)
 6040 FORMAT(1X,'     OUTPUT MATRIX',/,/,&
             1X,5X,A)
      END
```

(d) Output results

```
***  DAM1TM  ***

 **  INPUT  **

     LNA=11   LNB=11   LMC=11

     NM = 5   NN = 5   NL = 4

     ISW= 0

     INPUT MATRIX

     MATRIX A
          1.0    1.0    1.0    1.0    1.0
          2.0    2.0    2.0    2.0    2.0
          3.0    3.0    3.0    3.0    3.0
          4.0    4.0    4.0    4.0    4.0
          5.0    5.0    5.0    5.0    5.0

     MATRIX B
          1.0    1.0    1.0    1.0
          2.0    2.0    2.0    2.0
          3.0    3.0    3.0    3.0
          4.0    4.0    4.0    4.0
          5.0    5.0    5.0    5.0

 **  OUTPUT  **

     IERR =    0

     OUTPUT MATRIX

     MATRIX C
         55.0   55.0   55.0   55.0
         55.0   55.0   55.0   55.0
         55.0   55.0   55.0   55.0
         55.0   55.0   55.0   55.0
         55.0   55.0   55.0   55.0
```

3.2.10 DAM1TT, RAM1TT Multiplying Real Matrices (Two-Dimensional Array Type) $(C = C \pm A^T B^T)$

(1) **Function**

Obtain the product of real matrices A and B $(C = [C\pm]A^T B^T)$.

(2) **Usage**

Double precision:

CALL DAM1TT (A, LNA, NM, NN, B, LLB, NL, C, LMC, ISW, IERR)

Single precision:

CALL RAM1TT (A, LNA, NM, NN, B, LLB, NL, C, LMC, ISW, IERR)

(3) **Arguments**

D:Double precision real	Z:Double precision complex	I: $\left\{\begin{array}{l}\text{INTEGER(4) as for 32bit Integer}\\ \text{INTEGER(8) as for 64bit Integer}\end{array}\right\}$
R:Single precision real	C:Single precision complex	

No.	Argument	Type	Size	Input/ Output	Contents
1	A	$\left\{\begin{array}{c}\text{D}\\ \text{R}\end{array}\right\}$	LNA, NM	Input	Real matrix A (two-dimensional array type).
2	LNA	I	1	Input	Adjustable dimension of array A.
3	NM	I	1	Input	Number of columns in matrix A (Number of rows in matrix C).
4	NN	I	1	Input	Number of rows in matrix A (Number of columns in matrix B).
5	B	$\left\{\begin{array}{c}\text{D}\\ \text{R}\end{array}\right\}$	LLB, NN	Input	Real matrix B (two-dimensional array type).
6	LLB	I	1	Input	Adjustable dimension of array B.
7	NL	I	1	Input	Number of rows in matrix B (Number of columns in matrix C).
8	C	$\left\{\begin{array}{c}\text{D}\\ \text{R}\end{array}\right\}$	LMC, NL	Input	Initial real matrix C (when ISW= ±1) (two-dimensional array type).
				Output	Product of real matrices $(C = [C\pm]A^T B^T)$.
9	LMC	I	1	Input	Adjustable dimension of array C.
10	ISW	I	1	Input	Processing switch. ISW = 1: Obtain $C = C + A^T B^T$. ISW = 0: Obtain $C = A^T B^T$. ISW = −1: Obtain $C = C - A^T B^T$.
11	IERR	I	1	Output	Error indicator

(4) **Restrictions**

(a) $0 < \mathrm{NM} \le \mathrm{LMC}$

(b) $0 < \mathrm{NN} \le \mathrm{LNA}$

(c) $0 < \mathrm{NL} \le \mathrm{LNB}$

(d) $\mathrm{ISW} \in \{0, 1, -1\}$

(5) **Error indicator**

IERR value	Meaning	Processing
0	Normal termination.	
1000	NN was equal to 1.	Processing continues.
3000	Restriction (a), (b) or (c) was not satisfied.	Processing is aborted.
3010	Restriction (d) was not satisfied.	

(6) **Notes**
None

(7) **Example**

(a) Problem

$$A = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 3 & 3 \\ 4 & 4 & 4 & 4 & 4 \\ 5 & 5 & 5 & 5 & 5 \end{bmatrix}$$

$$B = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 3 & 3 \\ 4 & 4 & 4 & 4 & 4 \end{bmatrix}$$

Obtain $C = A^T B^T$.

(b) Input data
Matrices A and B, LNA = 11, LLB = 11, LNC = 11, NM = 5, NN = 5, NL = 4 and ISW = 0.

(c) Main program

```
      PROGRAM BAM1TT
! *** EXAMPLE OF DAM1TT ***
      IMPLICIT NONE
      INTEGER LNA,LLB,LMC,ISW,NM,NN,NL
      PARAMETER( LNA=11, LLB=11, LMC=11, ISW=0 )
      PARAMETER( NM=5, NN=5, NL=4 )
      INTEGER IERR,I,J
      REAL(8) A(LNA,NM),B(LLB,NN),C(LMC,NL)
!
      DO 100 I=1,NN
         READ(5,*) (A(I,J),J=1,NM)
  100 CONTINUE
      DO 110 I=1,NL
         READ(5,*) (B(I,J),J=1,NN)
  110 CONTINUE
!
      WRITE(6,6000) LNA,LLB,LMC,NM,NN,NL,ISW
      WRITE(6,6010) 'MATRIX A'
      DO 120 I=1,NN
         WRITE(6,6020) (A(I,J),J=1,NM)
  120 CONTINUE
      WRITE(6,6010) 'MATRIX B'
      DO 130 I=1,NL
```

```
         WRITE(6,6020) (B(I,J),J=1,NN)
  130 CONTINUE
!
      CALL DAM1TT(A,LNA,NM,NN,B,LLB,NL,C,LMC,ISW,IERR)
!
      WRITE(6,6030) IERR
      IF( IERR .GE. 3000 ) STOP
!
      WRITE(6,6040) 'MATRIX C'
      DO 140 I=1,NM
         WRITE(6,6020) (C(I,J),J=1,NL)
  140 CONTINUE
      STOP
!
 6000 FORMAT(/,&
             1X,'***  DAM1TT  ***',/,/,&
             1X,' **  INPUT  **',/,/,&
             1X,'     LNA=',I2,'   LLB=',I2,'   LMC=',I2,/,/,&
             1X,'     NM =',I2,'   NN =',I2,'   NL =',I2,/,/,&
             1X,'     ISW=',I2,/,/,&
             1X,'     INPUT MATRIX')
 6010 FORMAT(/,&
             1X,5X,A)
 6020 FORMAT(1X,6X,11(F7.1))
 6030 FORMAT(/,&
             1X,' **  OUTPUT  **',/,/,&
             1X,'     IERR = ',I4,/)
 6040 FORMAT(1X,'     OUTPUT MATRIX',/,/,&
             1X,5X,A)
      END
```

(d) Output results

```
***  DAM1TT  ***

 **  INPUT  **

    LNA=11   LLB=11   LMC=11

    NM = 5   NN = 5   NL = 4

    ISW= 0

    INPUT MATRIX

    MATRIX A
         1.0    1.0    1.0    1.0    1.0
         2.0    2.0    2.0    2.0    2.0
         3.0    3.0    3.0    3.0    3.0
         4.0    4.0    4.0    4.0    4.0
         5.0    5.0    5.0    5.0    5.0

    MATRIX B
         1.0    1.0    1.0    1.0    1.0
         2.0    2.0    2.0    2.0    2.0
         3.0    3.0    3.0    3.0    3.0
         4.0    4.0    4.0    4.0    4.0

 **  OUTPUT  **

    IERR =    0

    OUTPUT MATRIX

    MATRIX C
        15.0   30.0   45.0   60.0
        15.0   30.0   45.0   60.0
        15.0   30.0   45.0   60.0
        15.0   30.0   45.0   60.0
        15.0   30.0   45.0   60.0
```

3.2.11 ZAM1MM, CAM1MM Multiplying Complex Matrices (Two-Dimensional Array Type) (Real Argument Type) $(C = C \pm AB)$

(1) **Function**

Obtain the product of two complex matrices (Two-dimensional Array Type) ($C = [C\pm]AB$).

(2) **Usage**

Double precision:

CALL ZAM1MM (AR, AI, LMA, NM, NN, BR, BI, LNB, NL, CR, CI, LMC, ISW, IERR)

Single precision:

CALL CAM1MM (AR, AI, LMA, NM, NN, BR, BI, LNB, NL, CR, CI, LMC, ISW, IERR)

(3) **Arguments**

D:Double precision real Z:Double precision complex
R:Single precision real C:Single precision complex
$\text{I:}\left\{\begin{matrix}\text{INTEGER(4) as for 32bit Integer}\\ \text{INTEGER(8) as for 64bit Integer}\end{matrix}\right\}$

No.	Argument	Type	Size	Input/ Output	Contents
1	AR	$\left\{\begin{matrix}\text{D}\\ \text{R}\end{matrix}\right\}$	LMA, NN	Input	Real part of complex matrix A (two-dimensional array type).
2	AI	$\left\{\begin{matrix}\text{D}\\ \text{R}\end{matrix}\right\}$	LMA, NN	Input	Imaginary part of complex matrix A (two-dimensional array type).
3	LMA	I	1	Input	Adjustable dimension of arrays AR and AI.
4	NM	I	1	Input	Number of rows in matrix A (Number of rows in matrix C).
5	NN	I	1	Input	Number of columns in matrix A (Number of rows in matrix B).
6	BR	$\left\{\begin{matrix}\text{D}\\ \text{R}\end{matrix}\right\}$	LNB, NL	Input	Real part of complex matrix B (two-dimensional array type).
7	BI	$\left\{\begin{matrix}\text{D}\\ \text{R}\end{matrix}\right\}$	LNB, NL	Input	Imaginary part of complex matrix B (two-dimensional array type).
8	LNB	I	1	Input	Adjustable dimension of arrays BR and BI.
9	NL	I	1	Input	Number of columns in matrix B (Number of columns in matrix C).
10	CR	$\left\{\begin{matrix}\text{D}\\ \text{R}\end{matrix}\right\}$	LMC, NL	Input	Real part of initial complex matrix C (when ISW = ±1) (two-dimensional array type).
				Output	Product of complex matrices ($C = [C\pm]AB$).
11	CI	$\left\{\begin{matrix}\text{D}\\ \text{R}\end{matrix}\right\}$	LMC, NL	Input	Imaginary part of initial complex matrix C (when ISW = ±1) (two-dimensional array type).
				Output	Product of complex matrices ($C = [C\pm]AB$).

No.	Argument	Type	Size	Input/ Output	Contents
12	LMC	I	1	Input	Adjustable dimension of arrays CR and CI
13	ISW	I	1	Input	Processing switch. ISW = 1: Obtain $C = C + AB$ ISW = 0: Obtain $C = AB$ ISW = −1: Obtain $C = C - AB$
14	IERR	I	1	Output	Error indicator

(4) **Restrictions**

(a) $0 < \mathrm{NM} \leq \mathrm{LMA}, \mathrm{LMC}$

(b) $0 < \mathrm{NN} \leq \mathrm{LNB}$

(c) $\mathrm{NL} > 0$

(d) $\mathrm{ISW} \in \{0, 1, -1\}$

(5) **Error indicator**

IERR value	Meaning	Processing
0	Normal termination.	
1000	NN was equal to 1.	Processing continues.
3000	Restriction (a), (b) or (c) was not satisfied.	Processing is aborted.
3010	Restriction (d) was not satisfied.	

(6) **Notes**
None

(7) **Example**

(a) Problem

$$A = \begin{bmatrix} 1+i & 1+2i & 1+3i & 1+4i & 1+5i \\ 2+i & 2+2i & 2+3i & 2+4i & 2+5i \\ 3+i & 3+2i & 3+3i & 3+4i & 3+5i \\ 4+i & 4+2i & 4+3i & 4+4i & 4+5i \end{bmatrix}$$

$$B = \begin{bmatrix} 1+i & 1+2i & 1+3i & 1+4i \\ 2+i & 2+2i & 2+3i & 2+4i \\ 3+i & 3+2i & 3+3i & 3+4i \\ 4+i & 4+2i & 4+3i & 4+4i \\ 5+i & 5+2i & 5+3i & 5+4i \end{bmatrix}$$

Obtain $C = AB$.

(b) Input data
Matrices A and B, LMA = 11, LNB = 11, LNC = 11, NM = 4, NN = 5, NL = 4 and ISW = 0.

(c) Main program

```
      PROGRAM AAM1MM
! *** EXAMPLE OF ZAM1MM ***
      IMPLICIT NONE
      INTEGER LMA,LNB,LMC,ISW,NM,NN,NL
      PARAMETER( LMA=11, LNB=11, LMC=11, ISW=0 )
      PARAMETER( NM=4, NN=5, NL=4 )
      INTEGER IERR,I,J
      REAL(8) AR(LMA,NN),BR(LNB,NL),CR(LMC,NL)
      REAL(8) AI(LMA,NN),BI(LNB,NL),CI(LMC,NL)
!
      DO 100 I=1,NM
         READ(5,*) (AR(I,J),AI(I,J),J=1,NN)
  100 CONTINUE
      DO 110 I=1,NN
         READ(5,*) (BR(I,J),BI(I,J),J=1,NL)
  110 CONTINUE
!
      WRITE(6,6000) LMA,LNB,LMC,NM,NN,NL,ISW
      WRITE(6,6010) 'MATRIX A'
      DO 120 I=1,NM
         WRITE(6,6020) (AR(I,J),AI(I,J),J=1,NN)
  120 CONTINUE
      WRITE(6,6010) 'MATRIX B'
      DO 130 I=1,NN
         WRITE(6,6030) (BR(I,J),BI(I,J),J=1,NL)
  130 CONTINUE
!
      CALL ZAM1MM(AR,AI,LMA,NM,NN,BR,BI,LNB,NL,CR,CI,LMC,ISW,IERR)
!
      WRITE(6,6040) IERR
      IF( IERR .GE. 3000 ) STOP
!
      WRITE(6,6050) 'MATRIX C'
      DO 140 I=1,NM
         WRITE(6,6030) (CR(I,J),CI(I,J),J=1,NL)
  140 CONTINUE
      STOP
!
 6000 FORMAT(/,&
             1X,'***  ZAM1MM  ***',/,/,&
             1X,' **  INPUT  **',/,/,&
             1X,'     LMA=',I2,'   LNB=',I2,'   LMC=',I2,/,/,&
             1X,'     NM =',I2,'   NN =',I2,'   NL =',I2,/,/,&
             1X,'     ISW=',I2,/,/,&
             1X,'     INPUT MATRIX')
 6010 FORMAT(/,&
             1X,5X,A)
 6030 FORMAT(1X,6X,4('(',F5.1,',',F5.1,')'))
 6020 FORMAT(1X,6X,5('(',F5.1,',',F5.1,')'))
 6040 FORMAT(/,&
             1X,' **  OUTPUT  **',/,/,&
             1X,'     IERR = ',I4,/)
 6050 FORMAT(1X,'     OUTPUT MATRIX',/,/,&
             1X,5X,A)
      END
```

(d) Output results

```
***  ZAM1MM  ***

 **  INPUT  **

     LMA=11   LNB=11   LMC=11

     NM = 4   NN = 5   NL = 4

     ISW= 0

     INPUT MATRIX

     MATRIX A
      (  1.0,  1.0)(  1.0,  2.0)(  1.0,  3.0)(  1.0,  4.0)(  1.0,  5.0)
      (  2.0,  1.0)(  2.0,  2.0)(  2.0,  3.0)(  2.0,  4.0)(  2.0,  5.0)
      (  3.0,  1.0)(  3.0,  2.0)(  3.0,  3.0)(  3.0,  4.0)(  3.0,  5.0)
      (  4.0,  1.0)(  4.0,  2.0)(  4.0,  3.0)(  4.0,  4.0)(  4.0,  5.0)

     MATRIX B
      (  1.0,  1.0)(  1.0,  2.0)(  1.0,  3.0)(  1.0,  4.0)
      (  2.0,  1.0)(  2.0,  2.0)(  2.0,  3.0)(  2.0,  4.0)
      (  3.0,  1.0)(  3.0,  2.0)(  3.0,  3.0)(  3.0,  4.0)
      (  4.0,  1.0)(  4.0,  2.0)(  4.0,  3.0)(  4.0,  4.0)
      (  5.0,  1.0)(  5.0,  2.0)(  5.0,  3.0)(  5.0,  4.0)

 **  OUTPUT  **

     IERR =    0

     OUTPUT MATRIX

     MATRIX C
      (  0.0, 60.0)(-15.0, 65.0)(-30.0, 70.0)(-45.0, 75.0)
      ( 15.0, 65.0)(  0.0, 75.0)(-15.0, 85.0)(-30.0, 95.0)
      ( 30.0, 70.0)( 15.0, 85.0)(  0.0,100.0)(-15.0,115.0)
      ( 45.0, 75.0)( 30.0, 95.0)( 15.0,115.0)(  0.0,135.0)
```

3.2.12 ZAM1MH, CAM1MH Multiplying Complex Matrices (Two-Dimensional Array Type) (Real Argument Type) ($C = C \pm AB^*$)

(1) **Function**

Obtain the product of two complex matrices (Two-dimensional Array Type) ($C = [C\pm]AB^*$).

(2) **Usage**

Double precision:

CALL ZAM1MH (AR, AI, LMA, NM, NN, BR, BI, LLB, NL, CR, CI, LMC, ISW, IERR)

Single precision:

CALL CAM1MH (AR, AI, LMA, NM, NN, BR, BI, LLB, NL, CR, CI, LMC, ISW, IERR)

(3) **Arguments**

D:Double precision real Z:Double precision complex I: $\left\{\begin{matrix}\text{INTEGER(4) as for 32bit Integer}\\\text{INTEGER(8) as for 64bit Integer}\end{matrix}\right\}$
R:Single precision real C:Single precision complex

No.	Argument	Type	Size	Input/ Output	Contents
1	AR	$\begin{Bmatrix}\text{D}\\\text{R}\end{Bmatrix}$	LMA, NN	Input	Real part of complex matrix A (two-dimensional array type).
2	AI	$\begin{Bmatrix}\text{D}\\\text{R}\end{Bmatrix}$	LMA, NN	Input	Imaginary part of complex matrix A (two-dimensional array type).
3	LMA	I	1	Input	Adjustable dimension of arrays AR and AI.
4	NM	I	1	Input	Number of rows in matrix A (Number of rows in matrix C).
5	NN	I	1	Input	Number of columns in matrix A (Number of columns in matrix B).
6	BR	$\begin{Bmatrix}\text{D}\\\text{R}\end{Bmatrix}$	LLB, NN	Input	Real part of complex matrix B (two-dimensional array type).
7	BI	$\begin{Bmatrix}\text{D}\\\text{R}\end{Bmatrix}$	LLB, NN	Input	Real part of complex matrix B (two-dimensional array type).
8	LLB	I	1	Input	Adjustable dimension of arrays BR and BI.
9	NL	I	1	Input	Number of rows in matrix B (Number of columns in matrix C).
10	CR	$\begin{Bmatrix}\text{D}\\\text{R}\end{Bmatrix}$	LMC, NL	Input	Real part of initial complex matrix C (when ISW = ±1) (two-dimensional array type).
				Output	Product of complex matrices ($C = [C\pm]AB^*$).
11	CI	$\begin{Bmatrix}\text{D}\\\text{R}\end{Bmatrix}$	LMC, NL	Input	Imaginary part of initial complex matrix C (when ISW = ±1) (two-dimensional array type).
				Output	Product of complex matrices ($C = [C\pm]AB^*$).

No.	Argument	Type	Size	Input/ Output	Contents
12	LMC	I	1	Input	Adjustable dimension of arrays CR and CI.
13	ISW	I	1	Input	Processing switch. ISW = 1: Obtain $C = C + AB^*$ ISW = 0: Obtain $C = AB^*$ ISW = −1: Obtain $C = C - AB^*$
14	IERR	I	1	Output	Error indicator

(4) **Restrictions**

(a) $0 < \text{NM} \le \text{LMA}, \text{LMC}$

(b) $0 < \text{NL} \le \text{LLB}$

(c) $\text{NN} > 0$

(d) $\text{ISW} \in \{0, 1, -1\}$

(5) **Error indicator**

IERR value	Meaning	Processing
0	Normal termination.	
1000	NN was equal to 1.	Processing continues.
3000	Restriction (a), (b) or (c) was not satisfied.	Processing is aborted.
3010	Restriction (d) was not satisfied.	

(6) **Notes**
None

(7) **Example**

(a) Problem

$$A = \begin{bmatrix} 1+i & 1+2i & 1+3i & 1+4i & 1+5i \\ 2+i & 2+2i & 2+3i & 2+4i & 2+5i \\ 3+i & 3+2i & 3+3i & 3+4i & 3+5i \\ 4+i & 4+2i & 4+3i & 4+4i & 4+5i \end{bmatrix}$$

$$B = \begin{bmatrix} 1+i & 1+2i & 1+3i & 1+4i & 1+5i \\ 2+i & 2+2i & 2+3i & 2+4i & 2+5i \\ 3+i & 3+2i & 3+3i & 3+4i & 3+5i \\ 4+i & 4+2i & 4+3i & 4+4i & 4+5i \end{bmatrix}$$

Obtain $C = AB^*$.

(b) Input data
Matrices A and B, LMA = 11, LLB = 11, LNC = 11, NM = 4, NN = 5, NL = 4 and ISW = 0.

(c) Main program

```
      PROGRAM AAM1MH
! *** EXAMPLE OF ZAM1MH ***
      IMPLICIT NONE
      INTEGER LMA,LLB,LMC,ISW,NM,NN,NL
      PARAMETER( LMA=11, LLB=11, LMC=11, ISW=0 )
      PARAMETER( NM=4, NN=5, NL=4 )
      INTEGER IERR,I,J
      REAL(8) AR(LMA,NN),BR(LLB,NN),CR(LMC,NL)
      REAL(8) AI(LMA,NN),BI(LLB,NN),CI(LMC,NL)
!
      DO 100 I=1,NM
         READ(5,*) (AR(I,J),AI(I,J),J=1,NN)
  100 CONTINUE
      DO 110 I=1,NL
         READ(5,*) (BR(I,J),BI(I,J),J=1,NN)
  110 CONTINUE
!
      WRITE(6,6000) LMA,LLB,LMC,NM,NN,NL,ISW
      WRITE(6,6010) 'MATRIX A'
      DO 120 I=1,NM
         WRITE(6,6020) (AR(I,J),AI(I,J),J=1,NN)
  120 CONTINUE
      WRITE(6,6010) 'MATRIX B'
      DO 130 I=1,NL
         WRITE(6,6020) (BR(I,J),BI(I,J),J=1,NN)
  130 CONTINUE
!
      CALL ZAM1MH(AR,AI,LMA,NM,NN,BR,BI,LLB,NL,CR,CI,LMC,ISW,IERR)
!
      WRITE(6,6030) IERR
      IF( IERR .GE. 3000 ) STOP
!
      WRITE(6,6040) 'MATRIX C'
      DO 140 I=1,NM
         WRITE(6,6050) (CR(I,J),CI(I,J),J=1,NL)
  140 CONTINUE
      STOP
!
 6000 FORMAT(/,&
             1X,'***  ZAM1MH  ***',/,/,&
             1X,' **  INPUT  **',/,/,&
             1X,'     LMA=',I2,'   LLB=',I2,'   LMC=',I2,/,/,&
             1X,'     NM =',I2,'   NN =',I2,'   NL =',I2,/,/,&
             1X,'     ISW=',I2,/,/,&
             1X,'     INPUT MATRIX')
 6010 FORMAT(/,&
             1X,5X,A)
 6050 FORMAT(1X,6X,4('(',F5.1,',',F5.1,')'))
 6020 FORMAT(1X,6X,5('(',F5.1,',',F5.1,')'))
 6030 FORMAT(/,&
             1X,' **  OUTPUT  **',/,/,&
             1X,'     IERR = ',I4,/)
 6040 FORMAT(1X,'     OUTPUT MATRIX',/,/,&
             1X,5X,A)
      END
```

(d) Output results

```
***  ZAM1MH  ***

 **  INPUT  **

     LMA=11   LLB=11   LMC=11

     NM = 4   NN = 5   NL = 4

     ISW= 0

     INPUT MATRIX

     MATRIX A
      (  1.0,  1.0)(  1.0,  2.0)(  1.0,  3.0)(  1.0,  4.0)(  1.0,  5.0)
      (  2.0,  1.0)(  2.0,  2.0)(  2.0,  3.0)(  2.0,  4.0)(  2.0,  5.0)
      (  3.0,  1.0)(  3.0,  2.0)(  3.0,  3.0)(  3.0,  4.0)(  3.0,  5.0)
      (  4.0,  1.0)(  4.0,  2.0)(  4.0,  3.0)(  4.0,  4.0)(  4.0,  5.0)

     MATRIX B
      (  1.0,  1.0)(  1.0,  2.0)(  1.0,  3.0)(  1.0,  4.0)(  1.0,  5.0)
      (  2.0,  1.0)(  2.0,  2.0)(  2.0,  3.0)(  2.0,  4.0)(  2.0,  5.0)
      (  3.0,  1.0)(  3.0,  2.0)(  3.0,  3.0)(  3.0,  4.0)(  3.0,  5.0)
      (  4.0,  1.0)(  4.0,  2.0)(  4.0,  3.0)(  4.0,  4.0)(  4.0,  5.0)

 **  OUTPUT  **

     IERR =    0

     OUTPUT MATRIX

     MATRIX C
      ( 60.0,  0.0)( 65.0, 15.0)( 70.0, 30.0)( 75.0, 45.0)
      ( 65.0,-15.0)( 75.0,  0.0)( 85.0, 15.0)( 95.0, 30.0)
      ( 70.0,-30.0)( 85.0,-15.0)(100.0,  0.0)(115.0, 15.0)
      ( 75.0,-45.0)( 95.0,-30.0)(115.0,-15.0)(135.0,  0.0)
```

3.2.13 ZAM1HM, CAM1HM Multiplying Complex Matrices (Two-Dimensional Array Type) (Real Argument Type) ($C = C \pm A^*B$)

(1) **Function**

Obtain the product of complex matrix A and complex matrix B ($C = [C\pm]A^*B$)

(2) **Usage**

Double precision:

CALL ZAM1HM (AR, AI, LNA, NM, NN, BR, BI, LNB, NL, CR, CI, LMC, ISW, IERR)

Single precision:

CALL CAM1HM (AR, AI, LNA, NM, NN, BR, BI, LNB, NL, CR, CI, LMC, ISW, IERR)

(3) **Arguments**

D:Double precision real Z:Double precision complex I: $\left\{\begin{matrix}\text{INTEGER(4) as for 32bit Integer}\\\text{INTEGER(8) as for 64bit Integer}\end{matrix}\right\}$
R:Single precision real C:Single precision complex

No.	Argument	Type	Size	Input/ Output	Contents
1	AR	$\left\{\begin{matrix}D\\R\end{matrix}\right\}$	LNA, NM	Input	Real part of complex matrix A (two-dimensional array type).
2	AI	$\left\{\begin{matrix}D\\R\end{matrix}\right\}$	LNA, NM	Input	Imaginary part of complex matrix A (two-dimensional array type).
3	LNA	I	1	Input	Adjustable dimension of arrays AR and AI.
4	NM	I	1	Input	Number of columns in matrix A (Number of rows in matrix C).
5	NN	I	1	Input	Number of rows in matrix A (Number of rows in matrix B).
6	BR	$\left\{\begin{matrix}D\\R\end{matrix}\right\}$	LNB, NL	Input	Real part of complex matrix B (two-dimensional array type).
7	BI	$\left\{\begin{matrix}D\\R\end{matrix}\right\}$	LNB, NL	Input	Imaginary part of complex matrix B (two-dimensional array type).
8	LNB	I	1	Input	Adjustable dimension of arrays BR and BI.
9	NL	I	1	Input	Number of columns in matrix B (Number of columns in matrix C).
10	CR	$\left\{\begin{matrix}D\\R\end{matrix}\right\}$	LMC, NL	Input	Real part of initial complex matrix C (when ISW = ±1) (two-dimensional array type).
				Output	Real part of product ($C = [C\pm]A^*B$).
11	CI	$\left\{\begin{matrix}D\\R\end{matrix}\right\}$	LMC, NL	Input	Imaginary part of initial complex matrix C (when ISW = ±1) (two-dimensional array type).
				Output	Imaginary part of product ($C = [C\pm]A^*B$).

No.	Argument	Type	Size	Input/ Output	Contents
12	LMC	I	1	Input	Adjustable dimension of arrays CR and CI.
13	ISW	I	1	Input	Processing switch. ISW = 1: Obtain $C = C + A^*B$ ISW = 0: Obtain $C = A^*B$ ISW = −1: Obtain $C = C - A^*B$
14	IERR	I	1	Output	Error indicator

(4) **Restrictions**

(a) $0 < \mathrm{NM} \leq \mathrm{LMC}$

(b) $0 < \mathrm{NN} \leq \mathrm{LNA}, \mathrm{LNB}$

(c) $\mathrm{NL} > 0$

(d) $\mathrm{ISW} \in \{0, 1, -1\}$

(5) **Error indicator**

IERR value	Meaning	Processing
0	Normal termination.	
1000	NN was equal to 1.	Processing continues.
3000	Restriction (a), (b) or (c) was not satisfied.	Processing is aborted.
3010	Restriction (d) was not satisfied.	

(6) **Notes**
None

(7) **Example**

(a) Problem

$$A = \begin{bmatrix} 1+i & 1+2i & 1+3i & 1+4i & 1+5i \\ 2+i & 2+2i & 2+3i & 2+4i & 2+5i \\ 3+i & 3+2i & 3+3i & 3+4i & 3+5i \\ 4+i & 4+2i & 4+3i & 4+4i & 4+5i \end{bmatrix}$$

$$B = \begin{bmatrix} 1+i & 1+2i & 1+3i & 1+4i \\ 2+i & 2+2i & 2+3i & 2+4i \\ 3+i & 3+2i & 3+3i & 3+4i \\ 4+i & 4+2i & 4+3i & 4+4i \end{bmatrix}$$

Obtain $C = A^*B$.

(b) Input data
Matrices A and B, LNA = 11, LNB = 11, LNC = 11, NM = 5, NN = 4, NL = 4 and ISW = 0.

(c) Main program

```
      PROGRAM AAM1HM
! *** EXAMPLE OF ZAM1HM ***
      IMPLICIT NONE
      INTEGER LNA,LNB,LMC,ISW,NM,NN,NL
      PARAMETER( LNA=11, LNB=11, LMC=11, ISW=0 )
      PARAMETER( NM=5, NN=4, NL=4 )
      INTEGER IERR,I,J
      REAL(8) AR(LNA,NM),BR(LNB,NL),CR(LMC,NL)
      REAL(8) AI(LNA,NM),BI(LNB,NL),CI(LMC,NL)
!
      DO 100 I=1,NN
         READ(5,*) (AR(I,J),AI(I,J),J=1,NM)
  100 CONTINUE
      DO 110 I=1,NN
         READ(5,*) (BR(I,J),BI(I,J),J=1,NL)
  110 CONTINUE
!
      WRITE(6,6000) LNA,LNB,LMC,NM,NN,NL,ISW
      WRITE(6,6010) 'MATRIX A'
      DO 120 I=1,NN
         WRITE(6,6020) (AR(I,J),AI(I,J),J=1,NM)
  120 CONTINUE
      WRITE(6,6010) 'MATRIX B'
      DO 130 I=1,NN
         WRITE(6,6030) (BR(I,J),BI(I,J),J=1,NL)
  130 CONTINUE
!
      CALL ZAM1HM(AR,AI,LNA,NM,NN,BR,BI,LNB,NL,CR,CI,LMC,ISW,IERR)
!
      WRITE(6,6040) IERR
      IF( IERR .GE. 3000 ) STOP
!
      WRITE(6,6050) 'MATRIX C'
      DO 140 I=1,NM
         WRITE(6,6030) (CR(I,J),CI(I,J),J=1,NL)
  140 CONTINUE
      STOP
!
 6000 FORMAT(/,&
             1X,'***  ZAM1HM  ***',/,/,&
             1X,' **  INPUT  **',/,/,&
             1X,'     LNA=',I2,'   LNB=',I2,'   LMC=',I2,/,/,&
             1X,'     NM =',I2,'   NN =',I2,'   NL =',I2,/,/,&
             1X,'     ISW=',I2,/,/,&
             1X,'     INPUT MATRIX')
 6010 FORMAT(/,&
             1X,5X,A)
 6030 FORMAT(1X,4X,4('(',F5.1,',',F5.1,')'))
 6020 FORMAT(1X,4X,5('(',F5.1,',',F5.1,')'))
 6040 FORMAT(/,&
             1X,' **  OUTPUT  **',/,/,&
             1X,'     IERR = ',I4,/)
 6050 FORMAT(1X,'     OUTPUT MATRIX',/,/,&
             1X,5X,A)
      END
```

(d) Output results

```
 ***  ZAM1HM  ***

  **  INPUT  **

      LNA=11   LNB=11   LMC=11

      NM = 5   NN = 4   NL = 4

      ISW= 0

      INPUT MATRIX

      MATRIX A
     (  1.0,  1.0)(  1.0,  2.0)(  1.0,  3.0)(  1.0,  4.0)(  1.0,  5.0)
     (  2.0,  1.0)(  2.0,  2.0)(  2.0,  3.0)(  2.0,  4.0)(  2.0,  5.0)
     (  3.0,  1.0)(  3.0,  2.0)(  3.0,  3.0)(  3.0,  4.0)(  3.0,  5.0)
     (  4.0,  1.0)(  4.0,  2.0)(  4.0,  3.0)(  4.0,  4.0)(  4.0,  5.0)

      MATRIX B
     (  1.0,  1.0)(  1.0,  2.0)(  1.0,  3.0)(  1.0,  4.0)
     (  2.0,  1.0)(  2.0,  2.0)(  2.0,  3.0)(  2.0,  4.0)
     (  3.0,  1.0)(  3.0,  2.0)(  3.0,  3.0)(  3.0,  4.0)
     (  4.0,  1.0)(  4.0,  2.0)(  4.0,  3.0)(  4.0,  4.0)

  **  OUTPUT  **

      IERR =    0

      OUTPUT MATRIX

      MATRIX C
     ( 34.0,  0.0)( 38.0, 10.0)( 42.0, 20.0)( 46.0, 30.0)
     ( 38.0,-10.0)( 46.0,  0.0)( 54.0, 10.0)( 62.0, 20.0)
     ( 42.0,-20.0)( 54.0,-10.0)( 66.0,  0.0)( 78.0, 10.0)
     ( 46.0,-30.0)( 62.0,-20.0)( 78.0,-10.0)( 94.0,  0.0)
     ( 50.0,-40.0)( 70.0,-30.0)( 90.0,-20.0)(110.0,-10.0)
```

3.2.14 ZAM1HH, CAM1HH Multiplying Complex Matrices (Two-Dimensional Array Type) (Real Argument Type) ($C = C \pm A^*B^*$)

(1) **Function**

Obtain the product of complex matrix A and complex matrix B ($C = [C\pm]A^*B^*$)

(2) **Usage**

Double precision:

CALL ZAM1HH (AR, AI, LNA, NM, NN, BR, BI, LLB, NL, CR, CI, LMC, ISW, IERR)

Single precision:

CALL CAM1HH (AR, AI, LNA, NM, NN, BR, BI, LLB, NL, CR, CI, LMC, ISW, IERR)

(3) **Arguments**

D:Double precision real Z:Double precision complex I: { INTEGER(4) as for 32bit Integer }
R:Single precision real C:Single precision complex { INTEGER(8) as for 64bit Integer }

No.	Argument	Type	Size	Input/ Output	Contents
1	AR	{D R}	LNA, NM	Input	Real part of complex matrix A (two-dimensional array).
2	AI	{D R}	LNA, NM	Input	Imaginary part of complex matrix A (two-dimensional array).
3	LNA	I	1	Input	Adjustable dimension of arrays AR and AI.
4	NM	I	1	Input	Number of columns in matrix A (Number of rows in matrix C).
5	NN	I	1	Input	Number of rows in matrix A (Number of columns in matrix B).
6	BR	{D R}	LLB, NN	Input	Real part of complex matrix A (two-dimensional array type).
7	BI	{D R}	LLB, NN	Input	Imaginary part of complex matrix B (two-dimensional array type).
8	LLB	I	1	Input	Adjustable dimension of arrays BR and BI.
9	NL	I	1	Input	Number of rows in matrix B (Number of columns in matrix C).
10	CR	{D R}	LMC, NL	Input	Real part of initial complex matrix C (when ISW = ±1) (two-dimensional array type).
				Output	Product of complex matrices ($C = [C\pm]A^*B^*$).
11	CI	{D R}	LMC, NL	Input	Imaginary part of initial complex matrix C (when ISW = ±1) (two-dimensional array type).
				Output	Product of complex matrices ($C = [C\pm]A^*B^*$).

No.	Argument	Type	Size	Input/ Output	Contents
12	LMC	I	1	Input	Adjustable dimension of arrays CR and CI.
13	ISW	I	1	Input	Processing switch. ISW = 1: Obtain $C = C + A^*B^*$ ISW = 0: Obtain $C = A^*B^*$ ISW = −1: Obtain $C = C - A^*B^*$
14	IERR	I	1	Output	Error indicator

(4) **Restrictions**

(a) $0 <$ NM $\leq$ LMC

(b) $0 <$ NN $\leq$ LNA

(c) $0 <$ NL $\leq$ LNB

(d) ISW $\in \{0, 1, -1\}$

(5) **Error indicator**

IERR value	Meaning	Processing
0	Normal termination.	
1000	NN was equal to 1.	Processing continues.
3000	Restriction (a), (b) or (c) was not satisfied.	Processing is aborted.
3010	Restriction (d) was not satisfied.	

(6) **Notes**
None

(7) **Example**

(a) Problem

$$A = \begin{bmatrix} 1+i & 1+2i & 1+3i & 1+4i & 1+5i \\ 2+i & 2+2i & 2+3i & 2+4i & 2+5i \\ 3+i & 3+2i & 3+3i & 3+4i & 3+5i \\ 4+i & 4+2i & 4+3i & 4+4i & 4+5i \\ 5+i & 5+2i & 5+3i & 5+4i & 5+5i \end{bmatrix}$$

$$B = \begin{bmatrix} 1+i & 1+2i & 1+3i & 1+4i & 1+5i \\ 2+i & 2+2i & 2+3i & 2+4i & 2+5i \\ 3+i & 3+2i & 3+3i & 3+4i & 3+5i \\ 4+i & 4+2i & 4+3i & 4+4i & 4+5i \end{bmatrix}$$

Obtain $C = A^*B^*$.

(b) Input data
Matrices A and B, LNA = 11, LLB = 11, LNC = 11, NM = 5, NN = 5, NL = 4 and ISW = 0.

(c) Main program

```
      PROGRAM AAM1HH
! *** EXAMPLE OF ZAM1HH ***
      IMPLICIT NONE
      INTEGER LNA,LLB,LMC,ISW,NM,NN,NL
      PARAMETER( LNA=11, LLB=11, LMC=11, ISW=0 )
      PARAMETER( NM=5, NN=5, NL=4 )
      INTEGER IERR,I,J
      REAL(8) AR(LNA,NM),BR(LLB,NN),CR(LMC,NL)
      REAL(8) AI(LNA,NM),BI(LLB,NN),CI(LMC,NL)
!
      DO 100 I=1,NN
         READ(5,*) (AR(I,J),AI(I,J),J=1,NM)
  100 CONTINUE
      DO 110 I=1,NL
         READ(5,*) (BR(I,J),BI(I,J),J=1,NN)
  110 CONTINUE
!
      WRITE(6,6000) LNA,LLB,LMC,NM,NN,NL,ISW
      WRITE(6,6010) 'MATRIX A'
      DO 120 I=1,NN
         WRITE(6,6020) (AR(I,J),AI(I,J),J=1,NM)
  120 CONTINUE
      WRITE(6,6010) 'MATRIX B'
      DO 130 I=1,NL
         WRITE(6,6020) (BR(I,J),BI(I,J),J=1,NN)
  130 CONTINUE
!
      CALL ZAM1HH(AR,AI,LNA,NM,NN,BR,BI,LLB,NL,CR,CI,LMC,ISW,IERR)
!
      WRITE(6,6030) IERR
      IF( IERR .GE. 3000 ) STOP
!
      WRITE(6,6040) 'MATRIX C'
      DO 140 I=1,NM
         WRITE(6,6050) (CR(I,J),CI(I,J),J=1,NL)
  140 CONTINUE
      STOP
!
 6000 FORMAT(/,&
             1X,'***  ZAM1HH  ***',/,/,&
             1X,' **  INPUT  **',/,/,&
             1X,'     LNA=',I2,'   LLB=',I2,'   LMC=',I2,/,/,&
             1X,'     NM =',I2,'   NN =',I2,'   NL =',I2,/,/,&
             1X,'     ISW=',I2,/,/,&
             1X,'     INPUT MATRIX')
 6010 FORMAT(/,&
             1X,5X,A)
 6020 FORMAT(1X,4X,5('(',F5.1,',',F5.1,')'))
 6050 FORMAT(1X,4X,4('(',F6.1,',',F6.1,')'))
 6030 FORMAT(/,&
             1X,' **  OUTPUT  **',/,/,&
             1X,'     IERR = ',I4,/)
 6040 FORMAT(1X,'     OUTPUT MATRIX',/,/,&
             1X,5X,A)
      END
```

(d) Output results

```
 ***  ZAM1HH  ***

  **  INPUT  **

      LNA=11   LLB=11   LMC=11

      NM = 5   NN = 5   NL = 4

      ISW= 0

      INPUT MATRIX

      MATRIX A
     (  1.0,  1.0)(  1.0,  2.0)(  1.0,  3.0)(  1.0,  4.0)(  1.0,  5.0)
     (  2.0,  1.0)(  2.0,  2.0)(  2.0,  3.0)(  2.0,  4.0)(  2.0,  5.0)
     (  3.0,  1.0)(  3.0,  2.0)(  3.0,  3.0)(  3.0,  4.0)(  3.0,  5.0)
     (  4.0,  1.0)(  4.0,  2.0)(  4.0,  3.0)(  4.0,  4.0)(  4.0,  5.0)
     (  5.0,  1.0)(  5.0,  2.0)(  5.0,  3.0)(  5.0,  4.0)(  5.0,  5.0)

      MATRIX B
     (  1.0,  1.0)(  1.0,  2.0)(  1.0,  3.0)(  1.0,  4.0)(  1.0,  5.0)
     (  2.0,  1.0)(  2.0,  2.0)(  2.0,  3.0)(  2.0,  4.0)(  2.0,  5.0)
     (  3.0,  1.0)(  3.0,  2.0)(  3.0,  3.0)(  3.0,  4.0)(  3.0,  5.0)
     (  4.0,  1.0)(  4.0,  2.0)(  4.0,  3.0)(  4.0,  4.0)(  4.0,  5.0)

  **  OUTPUT  **

      IERR =    0

      OUTPUT MATRIX

      MATRIX C
     (   0.0, -60.0)(  15.0, -65.0)(  30.0, -70.0)(  45.0, -75.0)
     ( -15.0, -65.0)(   0.0, -75.0)(  15.0, -85.0)(  30.0, -95.0)
     ( -30.0, -70.0)( -15.0, -85.0)(   0.0,-100.0)(  15.0,-115.0)
     ( -45.0, -75.0)( -30.0, -95.0)( -15.0,-115.0)(   0.0,-135.0)
     ( -60.0, -80.0)( -45.0,-105.0)( -30.0,-130.0)( -15.0,-155.0)
```

3.2.15 ZAN1MM, CAN1MM Multiplying Complex Matrices (Two-Dimensional Array Type) (Complex Argument Type) $(C = C \pm AB)$

(1) **Function**

Obtain the product of two complex matrices (Two-dimensional Array Type) $(C = [C\pm]AB)$.

(2) **Usage**

Double precision:

CALL ZAN1MM (A, LMA, NM, NN, B, LNB, NL, C, LMC, ISW, IERR)

Single precision:

CALL CAN1MM (A, LMA, NM, NN, B, LNB, NL, C, LMC, ISW, IERR)

(3) **Arguments**

D:Double precision real Z:Double precision complex I: { INTEGER(4) as for 32bit Integer
R:Single precision real C:Single precision complex { INTEGER(8) as for 64bit Integer }

No.	Argument	Type	Size	Input/ Output	Contents
1	A	D R	LMA, NN	Input	Complex matrix A (two-dimensional array type).
2	LMA	I	1	Input	Adjustable dimension of array A.
3	NM	I	1	Input	Number of rows in matrix A (Number of rows in matrix C).
4	NN	I	1	Input	Number of columns in matrix A (Number of rows in matrix B).
5	B	D R	LNB, NL	Input	Complex matrix B (two-dimensional array type).
6	LNB	I	1	Input	Adjustable dimension of array B.
7	NL	I	1	Input	Number of columns in matrix B (Number of columns in matrix C).
8	C	D R	LMC, NL	Input	Initial complex matrix C (when ISW $= \pm 1$) (two-dimensional array type)
				Output	Product of complex matrices $(C = [C\pm]AB)$.
9	LMC	I	1	Input	Adjustable dimension of array C.
10	ISW	I	1	Input	Processing switch. ISW = 1: Obtain $C = C + AB$ ISW = 0: Obtain $C = AB$ ISW = −1: Obtain $C = C - AB$
11	IERR	I	1	Output	Error indicator

(4) **Restrictions**

(a) $0 < \mathrm{NM} \leq \mathrm{LMA}, \mathrm{LMC}$

(b) $0 < \mathrm{NN} \leq \mathrm{LNB}$

(c) $\mathrm{NL} > 0$

(d) $\mathrm{ISW} \in \{0, 1, -1\}$

(5) **Error indicator**

IERR value	Meaning	Processing
0	Normal termination.	
1000	NN was equal to 1.	Processing continues.
3000	Restriction (a), (b) or (c) was not satisfied.	Processing is aborted.
3010	Restriction (d) was not satisfied.	

(6) **Notes**
None

(7) **Example**

(a) Problem

$$A = \begin{bmatrix} 1+i & 1+2i & 1+3i & 1+4i & 1+5i \\ 2+i & 2+2i & 2+3i & 2+4i & 2+5i \\ 3+i & 3+2i & 3+3i & 3+4i & 3+5i \\ 4+i & 4+2i & 4+3i & 4+4i & 4+5i \end{bmatrix}$$

$$B = \begin{bmatrix} 1+i & 1+2i & 1+3i & 1+4i \\ 2+i & 2+2i & 2+3i & 2+4i \\ 3+i & 3+2i & 3+3i & 3+4i \\ 4+i & 4+2i & 4+3i & 4+4i \\ 5+i & 5+2i & 5+3i & 5+4i \end{bmatrix}$$

Obtain $C = AB$.

(b) Input data
Matrices A and B, LMA = 11, LNB = 11, LNC = 11, NM = 4, NN = 5, NL = 4 and ISW = 0.

(c) Main program

```
      PROGRAM AAN1MM
! *** EXAMPLE OF ZAN1MM ***
      IMPLICIT NONE
      INTEGER LMA,LNB,LMC,ISW,NM,NN,NL
      PARAMETER( LMA=11, LNB=11, LMC=11, ISW=0 )
      PARAMETER( NM=4, NN=5, NL=4 )
      INTEGER IERR,I,J
      COMPLEX(8) A(LMA,NN),B(LNB,NL),C(LMC,NL)
!
      DO 100 I=1,NM
         READ(5,*) (A(I,J),J=1,NN)
  100 CONTINUE
      DO 110 I=1,NN
         READ(5,*) (B(I,J),J=1,NL)
  110 CONTINUE
!
      WRITE(6,6000) LMA,LNB,LMC,NM,NN,NL,ISW
      WRITE(6,6010) 'MATRIX A'
      DO 120 I=1,NM
         WRITE(6,6020) (A(I,J),J=1,NN)
  120 CONTINUE
      WRITE(6,6010) 'MATRIX B'
      DO 130 I=1,NN
```

```
         WRITE(6,6030) (B(I,J),J=1,NL)
  130 CONTINUE
!
      CALL ZAN1MM(A,LMA,NM,NN,B,LNB,NL,C,LMC,ISW,IERR)
!
      WRITE(6,6040) IERR
      IF( IERR .GE. 3000 ) STOP
!
      WRITE(6,6050) 'MATRIX C'
      DO 140 I=1,NM
         WRITE(6,6030) (C(I,J),J=1,NL)
  140 CONTINUE
      STOP
!
 6000 FORMAT(/,&
             1X,'***  ZAN1MM  ***',/,/,&
             1X,' **  INPUT  **',/,/,&
             1X,'     LMA=',I2,'   LNB=',I2,'   LMC=',I2,/,/,&
             1X,'     NM =',I2,'   NN =',I2,'   NL =',I2,/,/,&
             1X,'     ISW=',I2,/,/,&
             1X,'     INPUT MATRIX')
 6010 FORMAT(/,&
             1X,5X,A)
 6030 FORMAT(1X,6X,4('(',F5.1,',',F5.1,')'))
 6020 FORMAT(1X,6X,5('(',F5.1,',',F5.1,')'))
 6040 FORMAT(/,&
             1X,' **  OUTPUT  **',/,/,&
             1X,'     IERR = ',I4,/)
 6050 FORMAT(1X,'     OUTPUT MATRIX',/,/,&
             1X,5X,A)
      END
```

(d) Output results

```
***  ZAN1MM  ***

 **  INPUT  **

     LMA=11   LNB=11   LMC=11

     NM = 4   NN = 5   NL = 4

     ISW= 0

     INPUT MATRIX

     MATRIX A
      (  1.0,  1.0)(  1.0,  2.0)(  1.0,  3.0)(  1.0,  4.0)(  1.0,  5.0)
      (  2.0,  1.0)(  2.0,  2.0)(  2.0,  3.0)(  2.0,  4.0)(  2.0,  5.0)
      (  3.0,  1.0)(  3.0,  2.0)(  3.0,  3.0)(  3.0,  4.0)(  3.0,  5.0)
      (  4.0,  1.0)(  4.0,  2.0)(  4.0,  3.0)(  4.0,  4.0)(  4.0,  5.0)

     MATRIX B
      (  1.0,  1.0)(  1.0,  2.0)(  1.0,  3.0)(  1.0,  4.0)
      (  2.0,  1.0)(  2.0,  2.0)(  2.0,  3.0)(  2.0,  4.0)
      (  3.0,  1.0)(  3.0,  2.0)(  3.0,  3.0)(  3.0,  4.0)
      (  4.0,  1.0)(  4.0,  2.0)(  4.0,  3.0)(  4.0,  4.0)
      (  5.0,  1.0)(  5.0,  2.0)(  5.0,  3.0)(  5.0,  4.0)

 **  OUTPUT  **

     IERR =    0

     OUTPUT MATRIX

     MATRIX C
      (  0.0, 60.0)(-15.0, 65.0)(-30.0, 70.0)(-45.0, 75.0)
      ( 15.0, 65.0)(  0.0, 75.0)(-15.0, 85.0)(-30.0, 95.0)
      ( 30.0, 70.0)( 15.0, 85.0)(  0.0,100.0)(-15.0,115.0)
      ( 45.0, 75.0)( 30.0, 95.0)( 15.0,115.0)(  0.0,135.0)
```

3.2.16 ZAN1MH, CAN1MH Multiplying Complex Matrices (Two-Dimensional Array Type) (Complex Argument Type) ($C = C \pm AB^*$)

(1) **Function**

Obtain the product of two complex matrices (Two-dimensional Array Type) ($C = [C\pm]AB^*$).

(2) **Usage**

Double precision:

CALL ZAN1MH (A, LMA, NM, NN, B, LLB, NL, C, LMC, ISW, IERR)

Single precision:

CALL CAN1MH (A, LMA, NM, NN, B, LLB, NL, C, LMC, ISW, IERR)

(3) **Arguments**

D:Double precision real	Z:Double precision complex	I: { INTEGER(4) as for 32bit Integer }
		{ INTEGER(8) as for 64bit Integer }

R:Single precision real	C:Single precision complex	

No.	Argument	Type	Size	Input/ Output	Contents
1	A	{D R}	LMA, NN	Input	Complex matrix A (two-dimensional array type).
2	LMA	I	1	Input	Adjustable dimension of array A.
3	NM	I	1	Input	Number of rows in matrix A (Number of rows in matrix C).
4	NN	I	1	Input	Number of columns in matrix A (Number of columns in matrix B).
5	B	{D R}	LLB, NN	Input	Complex matrix B (two-dimensional array type).
6	LLB	I	1	Input	Adjustable dimension of array B.
7	NL	I	1	Input	Number of rows in matrix B (Number of columns in matrix C).
8	C	{D R}	LMC, NL	Input	Initial complex matrix C (when ISW = ±1) (two-dimensional array type).
				Output	Product of complex matrices ($C = [C\pm]AB^*$).
9	LMC	I	1	Input	Adjustable dimension of array C.
10	ISW	I	1	Input	Processing switch. ISW = 1: Obtain $C = C + AB^*$ ISW = 0: Obtain $C = AB^*$ ISW = −1: Obtain $C = C - AB^*$
11	IERR	I	1	Output	Error indicator

(4) **Restrictions**

(a) $0 < \mathrm{NM} \leq \mathrm{LMA}, \mathrm{LMC}$

(b) $0 < \mathrm{NL} \leq \mathrm{LLB}$

(c) $\mathrm{NN} > 0$

(d) $\mathrm{ISW} \in \{0, 1, -1\}$

(5) **Error indicator**

IERR value	Meaning	Processing
0	Normal termination.	
1000	NN was equal to 1.	Processing continues.
3000	Restriction (a), (b) or (c) was not satisfied.	Processing is aborted.
3010	Restriction (d) was not satisfied.	

(6) **Notes**
None

(7) **Example**

(a) Problem

$$A = \begin{bmatrix} 1+i & 1+2i & 1+3i & 1+4i & 1+5i \\ 2+i & 2+2i & 2+3i & 2+4i & 2+5i \\ 3+i & 3+2i & 3+3i & 3+4i & 3+5i \\ 4+i & 4+2i & 4+3i & 4+4i & 4+5i \end{bmatrix}$$

$$B = \begin{bmatrix} 1+i & 1+2i & 1+3i & 1+4i & 1+5i \\ 2+i & 2+2i & 2+3i & 2+4i & 2+5i \\ 3+i & 3+2i & 3+3i & 3+4i & 3+5i \\ 4+i & 4+2i & 4+3i & 4+4i & 4+5i \end{bmatrix}$$

Obtain $C = AB^*$.

(b) Input data

Matrices A and B, LMA = 11, LLB = 11, LNC = 11, NM = 4, NN = 5, NL = 4 and ISW = 0.

(c) Main program

```
      PROGRAM AAN1MH
! *** EXAMPLE OF ZAN1MH ***
      IMPLICIT NONE
      INTEGER LMA,LLB,LMC,ISW,NM,NN,NL
      PARAMETER( LMA=11, LLB=11, LMC=11, ISW=0 )
      PARAMETER( NM=4, NN=5, NL=4 )
      INTEGER IERR,I,J
      COMPLEX(8) A(LMA,NN),B(LLB,NN),C(LMC,NL)
!
      DO 100 I=1,NM
         READ(5,*) (A(I,J),J=1,NN)
  100 CONTINUE
      DO 110 I=1,NL
         READ(5,*) (B(I,J),J=1,NN)
  110 CONTINUE
!
      WRITE(6,6000) LMA,LLB,LMC,NM,NN,NL,ISW
      WRITE(6,6010) 'MATRIX A'
      DO 120 I=1,NM
         WRITE(6,6020) (A(I,J),J=1,NN)
  120 CONTINUE
      WRITE(6,6010) 'MATRIX B'
      DO 130 I=1,NL
         WRITE(6,6020) (B(I,J),J=1,NN)
  130 CONTINUE
```

```
!
      CALL ZAN1MH(A,LMA,NM,NN,B,LLB,NL,C,LMC,ISW,IERR)
!
      WRITE(6,6030) IERR
      IF( IERR .GE. 3000 ) STOP
!
      WRITE(6,6040) 'MATRIX C'
      DO 140 I=1,NM
         WRITE(6,6050) (C(I,J),J=1,NL)
  140 CONTINUE
      STOP
!
 6000 FORMAT(/,&
             1X,'***  ZAN1MH  ***',/,/,&
             1X,' **  INPUT  **',/,/,&
             1X,'     LMA=',I2,'   LLB=',I2,'   LMC=',I2,/,/,&
             1X,'     NM =',I2,'   NN =',I2,'   NL =',I2,/,/,&
             1X,'     ISW=',I2,/,/,&
             1X,'     INPUT MATRIX')
 6010 FORMAT(/,&
             1X,5X,A)
 6050 FORMAT(1X,6X,4('(',F5.1,',',F5.1,')'))
 6020 FORMAT(1X,6X,5('(',F5.1,',',F5.1,')'))
 6030 FORMAT(/,&
             1X,' **  OUTPUT  **',/,/,&
             1X,'     IERR = ',I4,/)
 6040 FORMAT(1X,'     OUTPUT MATRIX',/,/,&
             1X,5X,A)
      END
```

(d) Output results

```
***  ZAN1MH  ***

 **  INPUT  **

     LMA=11   LLB=11   LMC=11

     NM = 4   NN = 5   NL = 4

     ISW= 0

     INPUT MATRIX

     MATRIX A
      (  1.0,  1.0)(  1.0,  2.0)(  1.0,  3.0)(  1.0,  4.0)(  1.0,  5.0)
      (  2.0,  1.0)(  2.0,  2.0)(  2.0,  3.0)(  2.0,  4.0)(  2.0,  5.0)
      (  3.0,  1.0)(  3.0,  2.0)(  3.0,  3.0)(  3.0,  4.0)(  3.0,  5.0)
      (  4.0,  1.0)(  4.0,  2.0)(  4.0,  3.0)(  4.0,  4.0)(  4.0,  5.0)

     MATRIX B
      (  1.0,  1.0)(  1.0,  2.0)(  1.0,  3.0)(  1.0,  4.0)(  1.0,  5.0)
      (  2.0,  1.0)(  2.0,  2.0)(  2.0,  3.0)(  2.0,  4.0)(  2.0,  5.0)
      (  3.0,  1.0)(  3.0,  2.0)(  3.0,  3.0)(  3.0,  4.0)(  3.0,  5.0)
      (  4.0,  1.0)(  4.0,  2.0)(  4.0,  3.0)(  4.0,  4.0)(  4.0,  5.0)

 **  OUTPUT  **

     IERR =    0

     OUTPUT MATRIX

     MATRIX C
      ( 60.0,  0.0)( 65.0, 15.0)( 70.0, 30.0)( 75.0, 45.0)
      ( 65.0,-15.0)( 75.0,  0.0)( 85.0, 15.0)( 95.0, 30.0)
      ( 70.0,-30.0)( 85.0,-15.0)(100.0,  0.0)(115.0, 15.0)
      ( 75.0,-45.0)( 95.0,-30.0)(115.0,-15.0)(135.0,  0.0)
```

3.2.17 ZAN1HM, CAN1HM Multiplying Complex Matrices (Two-Dimensional Array Type) (Complex Argument Type) $(C = C \pm A^*B)$

(1) **Function**

Obtain the product of complex matrix A and complex matrix B $(C = [C\pm]A^*B)$

(2) **Usage**

Double precision:
CALL ZAN1HM (A, LNA, NM, NN, B, LNB, NL, C, LMC, ISW, IERR)
Single precision:
CALL CAN1HM (A, LNA, NM, NN, B, LNB, NL, C, LMC, ISW, IERR)

(3) **Arguments**

D:Double precision real Z:Double precision complex
R:Single precision real C:Single precision complex

I: $\left\{\begin{array}{l}\text{INTEGER(4) as for 32bit Integer}\\ \text{INTEGER(8) as for 64bit Integer}\end{array}\right\}$

No.	Argument	Type	Size	Input/ Output	Contents
1	A	$\left\{\begin{array}{c}\text{Z}\\ \text{C}\end{array}\right\}$	LNA, NM	Input	Complex matrix A (two-dimensional array type).
2	LNA	I	1	Input	Adjustable dimension of array A.
3	NM	I	1	Input	Number of columns in matrix A (Number of rows in matrix C).
4	NN	I	1	Input	Number of rows in matrix A (Number of rows in matrix B).
5	B	$\left\{\begin{array}{c}\text{D}\\ \text{R}\end{array}\right\}$	LNB, NL	Input	Real matrix B (two-dimensional array type).
6	LNB	I	1	Input	Adjustable dimension of array B.
7	NL	I	1	Input	Number of columns in matrix B (Number of columns in matrix C).
8	C	$\left\{\begin{array}{c}\text{Z}\\ \text{C}\end{array}\right\}$	LMC, NL	Input	Initial complex matrix C (when ISW $= \pm 1$) (two-dimensional array type).
				Output	Product of complex matrices $(C = [C\pm]A^*B)$.
9	LMC	I	1	Input	Adjustable dimension of array C.
10	ISW	I	1	Input	Processing switch. ISW = 1: Obtain $C = C + A^*B$ ISW = 0: Obtain $C = A^*B$ ISW = -1: Obtain $C = C - A^*B$
11	IERR	I	1	Output	Error indicator

(4) **Restrictions**

(a) $0 < \mathrm{NM} \leq \mathrm{LMC}$

(b) $0 < \mathrm{NN} \leq \mathrm{LNA}, \mathrm{LNB}$

(c) $\mathrm{NL} > 0$

(d) $\mathrm{ISW} \in \{0, 1, -1\}$

(5) **Error indicator**

IERR value	Meaning	Processing
0	Normal termination.	
1000	NN was equal to 1.	Processing continues.
3000	Restriction (a), (b) or (c) was not satisfied.	Processing is aborted.
3010	Restriction (d) was not satisfied.	

(6) **Notes**
None

(7) **Example**

(a) Problem

$$A = \begin{bmatrix} 1+i & 1+2i & 1+3i & 1+4i & 1+5i \\ 2+i & 2+2i & 2+3i & 2+4i & 2+5i \\ 3+i & 3+2i & 3+3i & 3+4i & 3+5i \\ 4+i & 4+2i & 4+3i & 4+4i & 4+5i \end{bmatrix}$$

$$B = \begin{bmatrix} 1+i & 1+2i & 1+3i & 1+4i \\ 2+i & 2+2i & 2+3i & 2+4i \\ 3+i & 3+2i & 3+3i & 3+4i \\ 4+i & 4+2i & 4+3i & 4+4i \end{bmatrix}$$

Obtain $C = A^*B$.

(b) Input data
Matrices A and B, LNA = 11, LNB = 11, LNC = 11, NM = 5, NN = 4, NL = 4 and ISW = 0.

(c) Main program

```
      PROGRAM AAN1HM
! *** EXAMPLE OF ZAN1HM ***
      IMPLICIT NONE
      INTEGER LNA,LNB,LMC,ISW,NM,NN,NL
      PARAMETER( LNA=11, LNB=11, LMC=11, ISW=0 )
      PARAMETER( NM=5, NN=4, NL=4 )
      INTEGER IERR,I,J
      COMPLEX(8) A(LNA,NM),B(LNB,NL),C(LMC,NL)
!
      DO 100 I=1,NN
         READ(5,*) (A(I,J),J=1,NM)
  100 CONTINUE
      DO 110 I=1,NN
         READ(5,*) (B(I,J),J=1,NL)
  110 CONTINUE
!
      WRITE(6,6000) LNA,LNB,LMC,NM,NN,NL,ISW
      WRITE(6,6010) 'MATRIX A'
      DO 120 I=1,NN
         WRITE(6,6020) (A(I,J),J=1,NM)
  120 CONTINUE
      WRITE(6,6010) 'MATRIX B'
      DO 130 I=1,NN
         WRITE(6,6030) (B(I,J),J=1,NL)
  130 CONTINUE
```

```
!
      CALL ZAN1HM(A,LNA,NM,NN,B,LNB,NL,C,LMC,ISW,IERR)
!
      WRITE(6,6040) IERR
      IF( IERR .GE. 3000 ) STOP
!
      WRITE(6,6050) 'MATRIX C'
      DO 140 I=1,NM
         WRITE(6,6030) (C(I,J),J=1,NL)
  140 CONTINUE
      STOP
!
 6000 FORMAT(/,&
             1X,'***  ZAN1HM  ***',/,/,&
             1X,' **  INPUT  **',/,/,&
             1X,'     LNA=',I2,'   LNB=',I2,'   LMC=',I2,/,/,&
             1X,'     NM =',I2,'   NN =',I2,'   NL =',I2,/,/,&
             1X,'     ISW=',I2,/,/,&
             1X,'     INPUT MATRIX')
 6010 FORMAT(/,&
             1X,5X,A)
 6030 FORMAT(1X,4X,4('(',F5.1,',',F5.1,')'))
 6020 FORMAT(1X,4X,5('(',F5.1,',',F5.1,')'))
 6040 FORMAT(/,&
             1X,' **  OUTPUT  **',/,/,&
             1X,'     IERR = ',I4,/)
 6050 FORMAT(1X,'     OUTPUT MATRIX',/,/,&
             1X,5X,A)
      END
```

(d) Output results

```
***  ZAN1HM  ***

 **  INPUT  **

     LNA=11   LNB=11   LMC=11

     NM = 5   NN = 4   NL = 4

     ISW= 0

     INPUT MATRIX

     MATRIX A
    (  1.0,  1.0)(  1.0,  2.0)(  1.0,  3.0)(  1.0,  4.0)(  1.0,  5.0)
    (  2.0,  1.0)(  2.0,  2.0)(  2.0,  3.0)(  2.0,  4.0)(  2.0,  5.0)
    (  3.0,  1.0)(  3.0,  2.0)(  3.0,  3.0)(  3.0,  4.0)(  3.0,  5.0)
    (  4.0,  1.0)(  4.0,  2.0)(  4.0,  3.0)(  4.0,  4.0)(  4.0,  5.0)

     MATRIX B
    (  1.0,  1.0)(  1.0,  2.0)(  1.0,  3.0)(  1.0,  4.0)
    (  2.0,  1.0)(  2.0,  2.0)(  2.0,  3.0)(  2.0,  4.0)
    (  3.0,  1.0)(  3.0,  2.0)(  3.0,  3.0)(  3.0,  4.0)
    (  4.0,  1.0)(  4.0,  2.0)(  4.0,  3.0)(  4.0,  4.0)

 **  OUTPUT  **

     IERR =    0

     OUTPUT MATRIX

     MATRIX C
    ( 34.0,  0.0)( 38.0, 10.0)( 42.0, 20.0)( 46.0, 30.0)
    ( 38.0,-10.0)( 46.0,  0.0)( 54.0, 10.0)( 62.0, 20.0)
    ( 42.0,-20.0)( 54.0,-10.0)( 66.0,  0.0)( 78.0, 10.0)
    ( 46.0,-30.0)( 62.0,-20.0)( 78.0,-10.0)( 94.0,  0.0)
    ( 50.0,-40.0)( 70.0,-30.0)( 90.0,-20.0)(110.0,-10.0)
```

3.2.18 ZAN1HH, CAN1HH Multiplying Complex Matrices (Two-Dimensional Array Type) (Complex Argument Type) $(C = C \pm A^*B^*)$

(1) **Function**

Obtain the product of complex matrix A and complex matrix B $(C = [C\pm]A^*B^*)$

(2) **Usage**

Double precision:

CALL ZAN1HH (A, LNA, NM, NN, B, LLB, NL, C, LMC, ISW, IERR)

Single precision:

CALL CAN1HH (A, LNA, NM, NN, B, LLB, NL, C, LMC, ISW, IERR)

(3) **Arguments**

D:Double precision real Z:Double precision complex I: $\left\{\begin{array}{l}\text{INTEGER(4) as for 32bit Integer}\\\text{INTEGER(8) as for 64bit Integer}\end{array}\right\}$
R:Single precision real C:Single precision complex

No.	Argument	Type	Size	Input/ Output	Contents
1	A	$\left\{\begin{array}{c}\text{Z}\\\text{C}\end{array}\right\}$	LNA, NM	Input	Complex matrix A (two-dimensional array type).
2	LNA	I	1	Input	Adjustable dimension of array A.
3	NM	I	1	Input	Number of columns in matrix A (Number of rows in matrix C).
4	NN	I	1	Input	Number of rows in matrix A (Number of columns in matrix B).
5	B	$\left\{\begin{array}{c}\text{Z}\\\text{C}\end{array}\right\}$	LLB, NN	Input	Complex matrix B (two-dimensional array type).
6	LLB	I	1	Input	Adjustable dimension of array B.
7	NL	I	1	Input	Number of rows in matrix B (Number of columns in matrix C).
8	C	$\left\{\begin{array}{c}\text{Z}\\\text{C}\end{array}\right\}$	LMC, NL	Input	Initial complex matrix C (when ISW $= \pm1$) (two-dimensional array type).
				Output	Product of real matrices $(C = [C\pm]A^*B^*)$.
9	LMC	I	1	Input	Adjustable dimension of array C.
10	ISW	I	1	Input	Processing switch. ISW $= 1$: Obtain $C = C + A^*B^*$ ISW $= 0$: Obtain $C = A^*B^*$ ISW $= -1$: Obtain $C = C - A^*B^*$
11	IERR	I	1	Output	Error indicator

(4) **Restrictions**

(a) $0 < \mathrm{NM} \leq \mathrm{LMC}$

(b) $0 < \mathrm{NN} \leq \mathrm{LNA}$

(c) $0 < \mathrm{NL} \leq \mathrm{LNB}$

(d) $\mathrm{ISW} \in \{0, 1, -1\}$

(5) **Error indicator**

IERR value	Meaning	Processing
0	Normal termination.	
1000	NN was equal to 1.	Processing continues.
3000	Restriction (a), (b) or (c) was not satisfied.	Processing is aborted.
3010	Restriction (d) was not satisfied.	

(6) **Notes**
None

(7) **Example**

(a) Problem

$$A = \begin{bmatrix} 1+i & 1+2i & 1+3i & 1+4i & 1+5i \\ 2+i & 2+2i & 2+3i & 2+4i & 2+5i \\ 3+i & 3+2i & 3+3i & 3+4i & 3+5i \\ 4+i & 4+2i & 4+3i & 4+4i & 4+5i \\ 5+i & 5+2i & 5+3i & 5+4i & 5+5i \end{bmatrix}$$

$$B = \begin{bmatrix} 1+i & 1+2i & 1+3i & 1+4i & 1+5i \\ 2+i & 2+2i & 2+3i & 2+4i & 2+5i \\ 3+i & 3+2i & 3+3i & 3+4i & 3+5i \\ 4+i & 4+2i & 4+3i & 4+4i & 4+5i \end{bmatrix}$$

Obtain $C = A^* B^*$.

(b) Input data
Matrices A and B, LNA = 11, LLB = 11, LNC = 11, NM = 5, NN = 5, NL = 4 and ISW = 0.

(c) Main program

```
      PROGRAM AAN1HH
! *** EXAMPLE OF ZAN1HH ***
      IMPLICIT NONE
      INTEGER LNA,LLB,LMC,ISW,NM,NN,NL
      PARAMETER( LNA=11, LLB=11, LMC=11, ISW=0 )
      PARAMETER( NM=5, NN=5, NL=4 )
      INTEGER IERR,I,J
      COMPLEX(8) A(LNA,NM),B(LLB,NN),C(LMC,NL)
!
      DO 100 I=1,NN
         READ(5,*) (A(I,J),J=1,NM)
  100 CONTINUE
      DO 110 I=1,NL
         READ(5,*) (B(I,J),J=1,NN)
  110 CONTINUE
!
      WRITE(6,6000) LNA,LLB,LMC,NM,NN,NL,ISW
      WRITE(6,6010) 'MATRIX A'
      DO 120 I=1,NN
         WRITE(6,6020) (A(I,J),J=1,NM)
  120 CONTINUE
      WRITE(6,6010) 'MATRIX B'
      DO 130 I=1,NL
```

```
         WRITE(6,6020) (B(I,J),J=1,NN)
  130 CONTINUE
!
      CALL ZAN1HH(A,LNA,NM,NN,B,LLB,NL,C,LMC,ISW,IERR)
!
      WRITE(6,6030) IERR
      IF( IERR .GE. 3000 ) STOP
!
      WRITE(6,6040) 'MATRIX C'
      DO 140 I=1,NM
         WRITE(6,6050) (C(I,J),J=1,NL)
  140 CONTINUE
      STOP
!
 6000 FORMAT(/,&
             1X,'***  ZAN1HH  ***',/,/,&
             1X,' **  INPUT  **',/,/,&
             1X,'     LNA=',I2,'   LLB=',I2,'   LMC=',I2,/,/,&
             1X,'     NM =',I2,'   NN =',I2,'   NL =',I2,/,/,&
             1X,'     ISW=',I2,/,/,&
             1X,'     INPUT MATRIX')
 6010 FORMAT(/,&
             1X,5X,A)
 6020 FORMAT(1X,4X,5('(',F5.1,',',F5.1,')'))
 6050 FORMAT(1X,4X,4('(',F6.1,',',F6.1,')'))
 6030 FORMAT(/,&
             1X,' **  OUTPUT  **',/,/,&
             1X,'     IERR = ',I4,/)
 6040 FORMAT(1X,'     OUTPUT MATRIX',/,/,&
             1X,5X,A)
      END
```

(d) Output results

```
***  ZAN1HH  ***

 **  INPUT  **

     LNA=11   LLB=11   LMC=11

     NM = 5   NN = 5   NL = 4

     ISW= 0

     INPUT MATRIX

     MATRIX A
    (  1.0,  1.0)(  1.0,  2.0)(  1.0,  3.0)(  1.0,  4.0)(  1.0,  5.0)
    (  2.0,  1.0)(  2.0,  2.0)(  2.0,  3.0)(  2.0,  4.0)(  2.0,  5.0)
    (  3.0,  1.0)(  3.0,  2.0)(  3.0,  3.0)(  3.0,  4.0)(  3.0,  5.0)
    (  4.0,  1.0)(  4.0,  2.0)(  4.0,  3.0)(  4.0,  4.0)(  4.0,  5.0)
    (  5.0,  1.0)(  5.0,  2.0)(  5.0,  3.0)(  5.0,  4.0)(  5.0,  5.0)

     MATRIX B
    (  1.0,  1.0)(  1.0,  2.0)(  1.0,  3.0)(  1.0,  4.0)(  1.0,  5.0)
    (  2.0,  1.0)(  2.0,  2.0)(  2.0,  3.0)(  2.0,  4.0)(  2.0,  5.0)
    (  3.0,  1.0)(  3.0,  2.0)(  3.0,  3.0)(  3.0,  4.0)(  3.0,  5.0)
    (  4.0,  1.0)(  4.0,  2.0)(  4.0,  3.0)(  4.0,  4.0)(  4.0,  5.0)

 **  OUTPUT  **

     IERR =    0

     OUTPUT MATRIX

     MATRIX C
    (   0.0, -60.0)(  15.0, -65.0)(  30.0, -70.0)(  45.0, -75.0)
    ( -15.0, -65.0)(   0.0, -75.0)(  15.0, -85.0)(  30.0, -95.0)
    ( -30.0, -70.0)( -15.0, -85.0)(   0.0,-100.0)(  15.0,-115.0)
    ( -45.0, -75.0)( -30.0, -95.0)( -15.0,-115.0)(   0.0,-135.0)
    ( -60.0, -80.0)( -45.0,-105.0)( -30.0,-130.0)( -15.0,-155.0)
```

3.2.19 DAM1VM, RAM1VM Multiplying a Real Matrix (Two-Dimensional Array Type) and a Vector

(1) **Function**

Obtain the product of the real matrix A (two-dimensional array type) and the vector $\boldsymbol{x}$.

(2) **Usage**

Double precision:
CALL DAM1VM (A, LMA, M, N, X, Y, IERR)
Single precision:
CALL RAM1VM (A, LMA, M, N, X, Y, IERR)

(3) **Arguments**

D:Double precision real Z:Double precision complex
R:Single precision real C:Single precision complex
I: $\left\{\begin{array}{l}\text{INTEGER(4) as for 32bit Integer}\\\text{INTEGER(8) as for 64bit Integer}\end{array}\right\}$

No.	Argument	Type	Size	Input/ Output	Contents
1	A	$\left\{\begin{array}{c}\text{D}\\\text{R}\end{array}\right\}$	LMA, N	Input	Real matrix A (two-dimensional array type).
2	LMA	I	1	Input	Adjustable dimension of array A.
3	M	I	1	Input	Number of rows in matrix A.
4	N	I	1	Input	Number of columns in matrix A.
5	X	$\left\{\begin{array}{c}\text{D}\\\text{R}\end{array}\right\}$	N	Input	Multiplier vector $\boldsymbol{x}$.
6	Y	$\left\{\begin{array}{c}\text{D}\\\text{R}\end{array}\right\}$	M	Output	Product ($A\boldsymbol{x}$) of matrix A and vector $\boldsymbol{x}$.
7	IERR	I	1	Output	Error indicator

(4) **Restrictions**

(a) N > 0

(b) 0 < M ≤ LMA

(5) **Error indicator**

IERR value	Meaning	Processing
0	Normal termination.	
1000	N was equal to 1.	Processing continues.
3000	Restriction (a) or (b) was not satisfied.	Processing is aborted.

(6) **Notes**

None

(7) Example

(a) Problem

$$A = \begin{bmatrix} 9 & 8 & 7 \\ 8 & 8 & 7 \\ 7 & 7 & 7 \\ 7 & 6 & 6 \\ 6 & 6 & 6 \\ 5 & 6 & 7 \end{bmatrix}$$

$$\boldsymbol{x} = \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix}$$

Obtain $\boldsymbol{y}= A\boldsymbol{x}$.

(b) Input data

Matrix A, vector $\boldsymbol{x}$, LMA = 11, M = 6 and N = 3.

(c) Main program

```
      PROGRAM BAM1VM
! *** EXAMPLE OF DAM1VM ***
      IMPLICIT NONE
      INTEGER LMA,M,N
      PARAMETER( LMA=11, M=6, N=3 )
      INTEGER IERR,I,J
      REAL(8) A(LMA,N),X(N),Y(M)
!
      DO 100 I=1,M
         READ(5,*) (A(I,J),J=1,N)
  100 CONTINUE
      DO 110 I=1,N
         READ(5,*) X(I)
  110 CONTINUE
!
      WRITE(6,6000) LMA,M,N
      DO 120 I=1,M
         WRITE(6,6010) (A(I,J),J=1,N)
  120 CONTINUE
      WRITE(6,6020)
      DO 130 I=1,N
         WRITE(6,6010) X(I)
  130 CONTINUE
!
      CALL DAM1VM(A,LMA,M,N,X,Y,IERR)
!
      WRITE(6,6030) IERR
      IF( IERR .GE. 3000 ) STOP
!
      WRITE(6,6040)
      DO 140 I=1,M
         WRITE(6,6010) Y(I)
  140 CONTINUE
      STOP
!
 6000 FORMAT(/,&
             1X,'***  DAM1VM  ***',/,/,&
             1X,' **  INPUT  **',/,/,&
             1X,'     LMA=',I2,'   M=',I2,'   N=',I2,/,/,&
             1X,'     INPUT MATRIX A',/)
 6010 FORMAT(1X,6X,11(F7.1))
 6020 FORMAT(/,&
             1X,'     INPUT VECTOR X',/)
 6030 FORMAT(/,&
             1X,' **  OUTPUT  **',/,/,&
             1X,'     IERR = ',I4,/)
 6040 FORMAT(1X,'     OUTPUT VECTOR Y',/)
      END
```

(d) Output results

```
***  DAM1VM  ***

 **  INPUT  **

     LMA=11   M= 6   N= 3

     INPUT MATRIX A

          9.0    8.0    7.0
          8.0    8.0    7.0
```

```
         7.0    7.0    7.0
         7.0    6.0    6.0
         6.0    6.0    6.0
         5.0    6.0    7.0

    INPUT VECTOR X

         1.0
        -1.0
         1.0

**  OUTPUT  **

    IERR =    0

    OUTPUT VECTOR Y

         8.0
         7.0
         7.0
         7.0
         6.0
         6.0
```

3.2.20 DAM3VM, RAM3VM
Multiplying a Real Band Matrix (Band Type) and a Vector

(1) **Function**

Obtain the product of the real band matrix A (band type) and the vector $\boldsymbol{x}$.

(2) **Usage**

Double precision:
CALL DAM3VM (A, LMA, N, MU, ML, X, Y, IERR)
Single precision:
CALL RAM3VM (A, LMA, N, MU, ML, X, Y, IERR)

(3) **Arguments**

D:Double precision real Z:Double precision complex
R:Single precision real C:Single precision complex
I: $\left\{ \begin{array}{l} \text{INTEGER(4) as for 32bit Integer} \\ \text{INTEGER(8) as for 64bit Integer} \end{array} \right\}$

No.	Argument	Type	Size	Input/ Output	Contents
1	A	$\left\{\begin{matrix}\text{D}\\\text{R}\end{matrix}\right\}$	LMA, N	Input	Real band matrix A (band type) (See Appendix B).
2	LMA	I	1	Input	Adjustable dimension of array A.
3	N	I	1	Input	Order of matrix A.
4	MU	I	1	Input	Upper band width of matrix A.
5	ML	I	1	Input	Lower band width of matrix A.
6	X	$\left\{\begin{matrix}\text{D}\\\text{R}\end{matrix}\right\}$	N	Input	Multiplier vector $\boldsymbol{x}$.
7	Y	$\left\{\begin{matrix}\text{D}\\\text{R}\end{matrix}\right\}$	N	Output	Product ($A\boldsymbol{x}$) of matrix A and vector $\boldsymbol{x}$.
8	IERR	I	1	Output	Error indicator

(4) **Restrictions**

(a) $\text{N} > 0$

(b) $0 \leq \text{MU} < \text{N}$

(c) $0 \leq \text{ML} < \text{N}$

(d) $\text{MU} + \text{ML} < \text{LMA}$

(5) Error indicator

IERR value	Meaning	Processing
0	Normal termination.	
1000	N was equal to 1.	Processing continues.
3000	Restriction (a), (b), (c) or (d) was not satisfied.	Processing is aborted.

(6) Notes

None

(7) Example

(a) Problem

$$A = \begin{bmatrix} 1 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 2 \end{bmatrix}$$

$$\boldsymbol{x} = \begin{bmatrix} 4 \\ 3 \\ 2 \\ 1 \end{bmatrix}$$

Obtain $\boldsymbol{y}= A\boldsymbol{x}$.

(b) Input data

Matrix A, Vector $\boldsymbol{x}$, LMA = 11, N = 4, MU = 1 and ML = 1.

(c) Main program

```
      PROGRAM BAM3VM
! *** EXAMPLE OF DAM3VM ***
      IMPLICIT NONE
      INTEGER LMA,N,MU,ML
      PARAMETER( LMA=11, N=4, MU=1, ML=1 )
      INTEGER LNA
      PARAMETER( LNA=11 )
      INTEGER IERR,I,J,JERR
      REAL(8) A(LMA,N),X(N),Y(N)
      REAL(8) AA(LNA,N)
!
      DO 100 I=1,N
         READ(5,*) (AA(I,J),J=1,N)
  100 CONTINUE
      DO 110 I=1,N
         READ(5,*) X(I)
  110 CONTINUE
!
      WRITE(6,6000) LMA,N,MU,ML
      DO 120 I=1,N
         WRITE(6,6010) (AA(I,J),J=1,N)
  120 CONTINUE
      WRITE(6,6020)
      DO 130 I=1,N
         WRITE(6,6010) X(I)
  130 CONTINUE
!
      CALL DABMCS(AA,LNA,N,MU,ML,A,LMA,JERR)
      IF( JERR .GE. 3000 ) THEN
         WRITE(6,6030) JERR
         STOP
      ENDIF
!
      CALL DAM3VM(A,LMA,N,MU,ML,X,Y,IERR)
!
      WRITE(6,6040) IERR
      IF( IERR .GE. 3000 ) STOP
!
      WRITE(6,6050)
      DO 140 I=1,N
         WRITE(6,6010) Y(I)
  140 CONTINUE
```

```
      STOP
!
 6000 FORMAT(/,&
             1X,'***  DAM3VM  ***',/,/,&
             1X,' **  INPUT  **',/,/,&
             1X,'     LMA=',I2,'   N=',I2,'   MU =',I2,'   ML =',I2,/,/,&
             1X,'     INPUT MATRIX A',/)
 6010 FORMAT(1X,6X,11(F7.1))
 6020 FORMAT(/,&
             1X,'     INPUT VECTOR X',/)
 6030 FORMAT(/,&
             1X,'     DABMCS IERR = ',I4,/)
 6040 FORMAT(/,&
             1X,' **  OUTPUT  **',/,/,&
             1X,'     IERR = ',I4,/)
 6050 FORMAT(1X,'     OUTPUT VECTOR Y',/)
      END
```

(d) Output results

```
***  DAM3VM  ***

 **  INPUT  **

     LMA=11   N= 4   MU = 1   ML = 1

     INPUT MATRIX A

         1.0   -1.0    0.0    0.0
        -1.0    2.0   -1.0    0.0
         0.0   -1.0    2.0   -1.0
         0.0    0.0   -1.0    2.0

     INPUT VECTOR X

         4.0
         3.0
         2.0
         1.0

 **  OUTPUT  **

     IERR =    0

     OUTPUT VECTOR Y

         1.0
         0.0
         0.0
         0.0
```

3.2.21 DAM4VM, RAM4VM Multiplying a Real Symmetric Band Matrix (Symmetric Band Type) and a Vector

(1) **Function**

Obtain the product of the real band symmetric matrix A (symmetric band type) and the vector $\boldsymbol{x}$.

(2) **Usage**

Double precision:
CALL DAM4VM (A, LMA, N, MB, X, Y, IERR)
Single precision:
CALL RAM4VM (A, LMA, N, MB, X, Y, IERR)

(3) **Arguments**

D:Double precision real Z:Double precision complex I: $\left\{\begin{array}{l}\text{INTEGER(4) as for 32bit Integer}\\\text{INTEGER(8) as for 64bit Integer}\end{array}\right\}$
R:Single precision real C:Single precision complex

No.	Argument	Type	Size	Input/ Output	Contents
1	A	$\begin{Bmatrix}\text{D}\\\text{R}\end{Bmatrix}$	LMA, N	Input	Real symmetric band matrix A (symmetric band type) (See Appendix B).
2	LMA	I	1	Input	Adjustable dimension of array A.
3	N	I	1	Input	Order of matrix A.
4	MB	I	1	Input	Band width of matrix A.
5	X	$\begin{Bmatrix}\text{D}\\\text{R}\end{Bmatrix}$	N	Input	Multiplier vector $\boldsymbol{x}$.
6	Y	$\begin{Bmatrix}\text{D}\\\text{R}\end{Bmatrix}$	N	Output	Product ($A\boldsymbol{x}$) of matrix A and vector $\boldsymbol{x}$.
7	IERR	I	1	Output	Error indicator

(4) **Restrictions**

(a) N $>$ 0

(b) 0 $\leq$ MB $<$ N

(c) MB $<$ LMA

(5) **Error indicator**

IERR value	Meaning	Processing
0	Normal termination.	
1000	N was equal to 1.	Processing continues.
3000	Restriction (a), (b) or (c) was not satisfied.	Processing is aborted.

(6) Notes

None

(7) Example

(a) Problem

$$A = \begin{bmatrix} 1 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & -2 \end{bmatrix}$$

$$\boldsymbol{x} = \begin{bmatrix} 4 \\ 3 \\ 2 \\ 1 \end{bmatrix}$$

Obtain $\boldsymbol{y}= A\boldsymbol{x}$.

(b) Input data

Matrix A, vector $\boldsymbol{x}$, LMA = 11, N = 4 and MB = 1.

(c) Main program

```
      PROGRAM BAM4VM
! *** EXAMPLE OF DAM4VM ***
      IMPLICIT NONE
      INTEGER LMA,N,MB
      PARAMETER( LMA=11, N=4, MB=1 )
      INTEGER LNA
      PARAMETER( LNA=11 )
      INTEGER IERR,I,J,JERR
      REAL(8) A(LMA,N),X(N),Y(N)
      REAL(8) AA(LNA,N)
!
      DO 100 I=1,N
         READ(5,*) (AA(I,J),J=1,N)
  100 CONTINUE
      DO 110 I=1,N
         READ(5,*) X(I)
  110 CONTINUE
!
      WRITE(6,6000) LMA,N,MB
      DO 120 I=1,N
         WRITE(6,6010) (AA(I,J),J=1,N)
  120 CONTINUE
      WRITE(6,6020)
      DO 130 I=1,N
         WRITE(6,6010) X(I)
  130 CONTINUE
!
      CALL DASBCS(AA,LNA,N,MB,A,LMA,JERR)
      IF( JERR .GE. 3000 ) THEN
         WRITE(6,6030) JERR
         STOP
      ENDIF
!
      CALL DAM4VM(A,LMA,N,MB,X,Y,IERR)
!
      WRITE(6,6040) IERR
      IF( IERR .GE. 3000 ) STOP
!
      WRITE(6,6050)
      DO 140 I=1,N
         WRITE(6,6010) Y(I)
  140 CONTINUE
      STOP
!
 6000 FORMAT(/,&
             1X,'***  DAM4VM  ***',/,/,&
             1X,' **  INPUT  **',/,/,&
             1X,'     LMA=',I2,'   N=',I2,'   MB=',I2,/,/,&
             1X,'     INPUT MATRIX A',/)
 6010 FORMAT(1X,6X,11(F7.1))
 6020 FORMAT(/,&
             1X,'     INPUT VECTOR X',/)
 6030 FORMAT(/,&
             1X,'     DASBCS IERR = ',I4,/)
 6040 FORMAT(/,&
             1X,' **  OUTPUT  **',/,/,&
             1X,'     IERR = ',I4,/)
 6050 FORMAT(1X,'     OUTPUT VECTOR Y',/)
      END
```

(d) Output results

```
***  DAM4VM  ***

 **  INPUT  **

     LMA=11   N= 4   MB= 1

     INPUT MATRIX A

          1.0   -1.0    0.0    0.0
         -1.0    2.0   -1.0    0.0
          0.0   -1.0    2.0   -1.0
          0.0    0.0   -1.0    2.0

     INPUT VECTOR X

          4.0
          3.0
          2.0
          1.0

 **  OUTPUT  **

     IERR =    0

     OUTPUT VECTOR Y

          1.0
          0.0
          0.0
          0.0
```

3.2.22 DAM1TP, RAM1TP
Transposing a Real Matrix (Two-Dimensional Array Type)

(1) **Function**
Obtain the transposed matrix of the real matrix A (two-dimensional array type).

(2) **Usage**
Double precision:
CALL DAM1TP (A, LMA, M, N, B, LNB, IERR)
Single precision:
CALL RAM1TP (A, LMA, M, N, B, LNB, IERR)

(3) **Arguments**

D:Double precision real Z:Double precision complex I: $\left\{\begin{array}{l}\text{INTEGER(4) as for 32bit Integer}\\ \text{INTEGER(8) as for 64bit Integer}\end{array}\right\}$
R:Single precision real C:Single precision complex

No.	Argument	Type	Size	Input/ Output	Contents
1	A	$\left\{\begin{matrix}\text{D}\\ \text{R}\end{matrix}\right\}$	LMA, N	Input	Real matrix A (two-dimensional array type).
2	LMA	I	1	Input	Adjustable dimension of array A.
3	M	I	1	Input	Number of rows in matrix A.
4	N	I	1	Input	Number of columns in matrix A.
5	B	$\left\{\begin{matrix}\text{D}\\ \text{R}\end{matrix}\right\}$	LNB, M	Output	Transposed matrix A^T (two-dimensional array type) of matrix A.
6	LNB	I	1	Input	Adjustable dimension of array B.
7	IERR	I	1	Output	Error indicator

(4) **Restrictions**

(a) $0 < \text{M} \leq \text{LMA}$

(b) $0 < \text{N} \leq \text{LNB}$

(5) **Error indicator**

IERR value	Meaning	Processing
0	Normal termination.	
1000	N was equal to 1.	Processing continues.
3000	Restriction (a) or (b) was not satisfied.	Processing is aborted.

(6) **Notes**
None

(7) **Example**

(a) Problem

$$A = \begin{bmatrix} 11 & 12 & 13 & 0 \\ 21 & 22 & 23 & 24 \\ 31 & 32 & 33 & 34 \\ 0 & 42 & 43 & 44 \end{bmatrix}$$

Obtain $B = A^T$.

(b) Input data

Matrix A, LMA = 11, LNB = 11 M = 4 and N = 4.

(c) Main program

```
      PROGRAM BAM1TP
! *** EXAMPLE OF DAM1TP ***
      IMPLICIT NONE
      INTEGER LMA,LNB,M,N
      PARAMETER( LMA=11, LNB=11, M=4, N=4 )
      INTEGER IERR,I,J
      REAL(8) A(LMA,N),B(LNB,M)
!
      DO 100 I=1,M
         READ(5,*) (A(I,J),J=1,N)
  100 CONTINUE
!
      WRITE(6,6000) LMA,LNB,M,N
      DO 110 I=1,M
         WRITE(6,6010) (A(I,J),J=1,N)
  110 CONTINUE
!
      CALL DAM1TP(A,LMA,M,N,B,LNB,IERR)
!
      WRITE(6,6020) IERR
      IF( IERR .GE. 3000 ) STOP
!
      WRITE(6,6030)
      DO 120 I=1,N
         WRITE(6,6010) (B(I,J),J=1,M)
  120 CONTINUE
      STOP
!
 6000 FORMAT(/,&
             1X,'***  DAM1TP  ***',/,/,&
             1X,' **  INPUT  **',/,/,&
             1X,'     LMA=',I2,'   LNB=',I2,'   M=',I2,'   N=',I2,/,/,&
             1X,'     INPUT MATRIX A',/)
 6010 FORMAT(1X,6X,11(F7.1))
 6020 FORMAT(/,&
             1X,' **  OUTPUT  **',/,/,&
             1X,'     IERR = ',I4,/)
 6030 FORMAT(1X,'     OUTPUT MATRIX B',/)
      END
```

(d) Output results

```
***  DAM1TP  ***

 **  INPUT  **

     LMA=11   LNB=11   M= 4   N= 4

     INPUT MATRIX A

         11.0   12.0   13.0   14.0
         21.0   22.0   23.0   24.0
         31.0   32.0   33.0   34.0
         41.0   42.0   43.0   44.0

 **  OUTPUT  **

     IERR =    0

     OUTPUT MATRIX B

         11.0   21.0   31.0   41.0
         12.0   22.0   32.0   42.0
         13.0   23.0   33.0   43.0
         14.0   24.0   34.0   44.0
```

3.2.23 DAM3TP, RAM3TP
Transposing a Real Band Matrix (Band Type)

(1) **Function**

Obtain the transposed matrix of the band matrix A (band type).

(2) **Usage**

Double precision:
  CALL DAM3TP (A, LMA, N, MU, ML, B, LMB, IERR)
Single precision:
  CALL RAM3TP (A, LMA, N, MU, ML, B, LMB, IERR)

(3) **Arguments**

D:Double precision real Z:Double precision complex
R:Single precision real C:Single precision complex

I: $\left\{\begin{array}{l}\text{INTEGER(4) as for 32bit Integer}\\ \text{INTEGER(8) as for 64bit Integer}\end{array}\right\}$

No.	Argument	Type	Size	Input/ Output	Contents
1	A	$\left\{\begin{array}{c}\text{D}\\ \text{R}\end{array}\right\}$	LMA, N	Input	Real band matrix A (band type) (See Appendix B).
2	LMA	I	1	Input	Adjustable dimension of array A.
3	N	I	1	Input	Order of matrix A.
4	MU	I	1	Input	Upper band width of matrix A.
5	ML	I	1	Input	Lower band width of matrix A.
6	B	$\left\{\begin{array}{c}\text{D}\\ \text{R}\end{array}\right\}$	LMB, N	Output	Transposed matrix A^T of matrix A (band type) (See Appendix B).
7	LMB	I	1	Input	Adjustable dimension of array B.
8	IERR	I	1	Output	Error indicator

(4) **Restrictions**

(a) $\text{N} > 0$

(b) $0 \leq \text{MU} < \text{N}$

(c) $0 \leq \text{ML} < \text{N}$

(d) $\text{MU} + \text{ML} < \text{LMA}, \text{LMB}$

(5) **Error indicator**

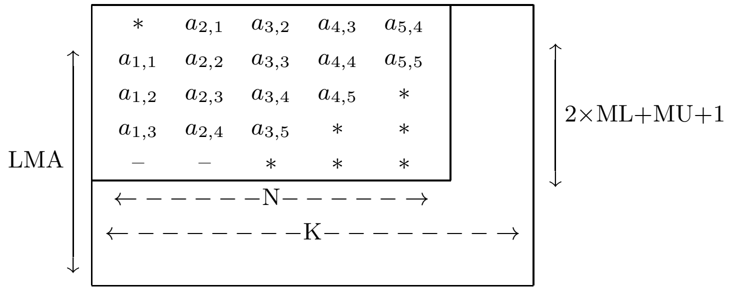
IERR value	Meaning	Processing
0	Normal termination.	
1000	N was equal to 1.	Processing continues.
3000	Restriction (a), (b), (c) or (d) was not satisfied.	Processing is aborted.

(6) Notes

(a) Array B elements that had not been stored in matrix A retain the values they had at the time the subroutine was called.

Example:

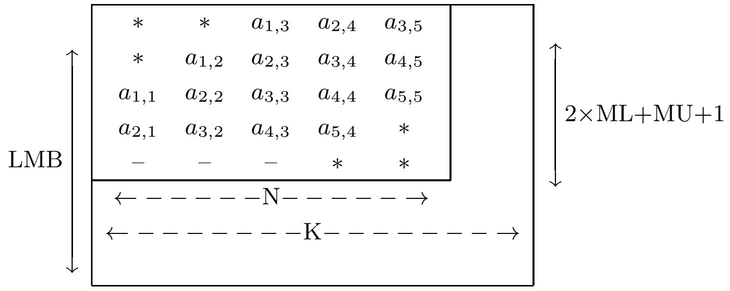
Storage status within array A(LMA, k)



$*$	$a_{2,1}$	$a_{3,2}$	$a_{4,3}$	$a_{5,4}$
$a_{1,1}$	$a_{2,2}$	$a_{3,3}$	$a_{4,4}$	$a_{5,5}$
$a_{1,2}$	$a_{2,3}$	$a_{3,4}$	$a_{4,5}$	$*$
$a_{1,3}$	$a_{2,4}$	$a_{3,5}$	$*$	$*$
–	–	$*$	$*$	$*$

⇓

Storage status within array B(LMB, k)



$*$	$*$	$a_{1,3}$	$a_{2,4}$	$a_{3,5}$
$*$	$a_{1,2}$	$a_{2,3}$	$a_{3,4}$	$a_{4,5}$
$a_{1,1}$	$a_{2,2}$	$a_{3,3}$	$a_{4,4}$	$a_{5,5}$
$a_{2,1}$	$a_{3,2}$	$a_{4,3}$	$a_{5,4}$	$*$
–	–	–	$*$	$*$

Remarks

a. Elements of B indicated by asterisks ($*$) and dashes (–) retain their input-time values.

b. The area indicated by dashes (–) is required for an LU decomposition of the matrix.

c. MU is the upper band width and ML is the lower band width.

d. LMB $>$ ML $+$ MU and K $\geq$ N must hold. (However, if LU decomposition is to be performed after conversion, LMB $\geq 2 \times$ ML $+$ MU $+ 1$ and K $\geq$ N must hold.)

(7) Example

(a) Problem

$$A = \begin{bmatrix} 11 & 12 & 13 & 0 \\ 21 & 22 & 23 & 24 \\ 0 & 32 & 33 & 34 \\ 0 & 0 & 43 & 44 \end{bmatrix}$$

Obtain $B = A^T$.

(b) Input data

Matrix A, LMA = 11, LMB = 11, N = 4, MU = 2 and ML = 1.

(c) Main program

```
      PROGRAM BAM3TP
! *** EXAMPLE OF DAM3TP ***
      IMPLICIT NONE
      INTEGER LMA,LMB,N,MU,ML
      PARAMETER( LMA=11, LMB=11, N=4, MU=2, ML=1 )
      INTEGER LNA
      PARAMETER( LNA=11 )
      INTEGER IERR,I,J,JERR
      REAL(8) A(LMA,N),B(LMB,N)
      REAL(8) AA(LNA,N)
!
      DO 100 I=1,N
         READ(5,*) (AA(I,J),J=1,N)
  100 CONTINUE
!
      WRITE(6,6000) LMA,LMB,N,MU,ML
      DO 110 I=1,N
         WRITE(6,6010) (AA(I,J),J=1,N)
  110 CONTINUE
!
      CALL DABMCS(AA,LNA,N,MU,ML,A,LMA,JERR)
      IF( JERR .GE. 3000 ) THEN
         WRITE(6,6020) JERR
         STOP
      ENDIF
!
      CALL DAM3TP(A,LMA,N,MU,ML,B,LMB,IERR)
!
      WRITE(6,6030) IERR
      IF( IERR .GE. 3000 ) STOP
!
      CALL DABMEL(B,LMB,N,ML,MU,AA,LNA,JERR)
      IF( JERR .GE. 3000 ) THEN
         WRITE(6,6040) JERR
         STOP
      ENDIF
!
      WRITE(6,6050)
      DO 120 I=1,N
         WRITE(6,6010) (AA(I,J),J=1,N)
  120 CONTINUE
      STOP
!
 6000 FORMAT(/,&
             1X,'***  DAM3TP  ***',/,/,&
             1X,' **  INPUT  **',/,/,&
             1X,'     LMA=',I2,'   LMB=',I2,/,/,&
             1X,'     N  =',I2,'   MU =',I2,'   ML =',I2,/,/,&
             1X,'     INPUT MATRIX A',/)
 6010 FORMAT(1X,6X,11(F7.1))
 6020 FORMAT(/,&
             1X,'     DABMCS IERR = ',I4,/)
 6030 FORMAT(/,&
             1X,' **  OUTPUT  **',/,/,&
             1X,'     IERR = ',I4,/)
 6040 FORMAT(/,&
             1X,'     DABMEL IERR = ',I4,/)
 6050 FORMAT(1X,'     OUTPUT MATRIX B',/)
      END
```

(d) Output results

```
***  DAM3TP  ***

 **  INPUT  **

     LMA=11   LMB=11

     N  = 4   MU = 2   ML = 1

     INPUT MATRIX A

         11.0   12.0   13.0    0.0
         21.0   22.0   23.0   24.0
          0.0   32.0   33.0   34.0
          0.0    0.0   43.0   44.0

 **  OUTPUT  **

     IERR =    0

     OUTPUT MATRIX B

         11.0   21.0    0.0    0.0
         12.0   22.0   32.0    0.0
         13.0   23.0   33.0   43.0
          0.0   24.0   34.0   44.0
```

3.2.24 DAMVJ1, RAMVJ1 Matrix–Vector Product of a Real Random Sparse Matrix (JAD; Jagged Diagonals Storage Type) ($\boldsymbol{y} = \beta\boldsymbol{y} + \alpha A\boldsymbol{x}$)

(1) **Function**

Obtain the product ($\boldsymbol{y} = \beta\boldsymbol{y} + \alpha A\boldsymbol{x}$) of real random sparse matrix A (JAD; jagged diagonals storage type) and vector $\boldsymbol{x}$.

(2) **Usage**

Double precision:

CALL DAMVJ1 (AJAD, LXA, IAJAD, JAJAD, JADORD, N, MJAD, ALPHA, BETA, X,
Y, W, IERR)

Single precision:

CALL RAMVJ1 (AJAD, LXA, IAJAD, JAJAD, JADORD, N, MJAD, ALPHA, BETA, X,
Y, W, IERR)

(3) **Arguments**

D:Double precision real Z:Double precision complex
R:Single precision real C:Single precision complex

I: $\left\{\begin{array}{l}\text{INTEGER(4) as for 32bit Integer}\\ \text{INTEGER(8) as for 64bit Integer}\end{array}\right\}$

No.	Argument	Type	Size	Input/ Output	Contents
1	AJAD	$\left\{\begin{array}{c}\text{D}\\ \text{R}\end{array}\right\}$	LXA	Input	Sparse matrix A (JAD storage type) (See Appendix B).
2	LXA	I	1	Input	Size allocated for arrays AJAD and JAJAD.
3	IAJAD	I	MJAD+1	Input	Array of indices for sparse matrix A (JAD storage type) (See Appendix B).
4	JAJAD	I	LXA	Input	Array of indices for sparse matrix A (JAD storage type) (See Appendix B).
5	JADORD	I	N	Input	Array of indices for sparse matrix A (JAD storage type) (See Appendix B).
6	N	I	1	Input	Order of vectors X and Y.
7	MJAD	I	1	Input	Number of jagged diagonals for JAD storage of matrix A.
8	ALPHA	$\left\{\begin{array}{c}\text{D}\\ \text{R}\end{array}\right\}$	1	Input	Multiplier α of $\boldsymbol{y} = \beta\boldsymbol{y} + \alpha A\boldsymbol{x}$.
9	BETA	$\left\{\begin{array}{c}\text{D}\\ \text{R}\end{array}\right\}$	1	Input	Multiplier β of $\boldsymbol{y} = \beta\boldsymbol{y} + \alpha A\boldsymbol{x}$.
10	X	$\left\{\begin{array}{c}\text{D}\\ \text{R}\end{array}\right\}$	N	Input	Vector $\boldsymbol{x}$ of $\boldsymbol{y} = \beta\boldsymbol{y} + \alpha A\boldsymbol{x}$.
11	Y	$\left\{\begin{array}{c}\text{D}\\ \text{R}\end{array}\right\}$	N	Input/ Output	Vector $\boldsymbol{y}$ of $\boldsymbol{y} = \beta\boldsymbol{y} + \alpha A\boldsymbol{x}$.

No.	Argument	Type	Size	Input/ Output	Contents
12	W	$\begin{Bmatrix}\text{D}\\\text{R}\end{Bmatrix}$	N	Work	Work area
13	IERR	I	1	Output	Error indicator

(4) **Restrictions**

(a) N > 0

(b) 0 < MJAD ≤ N

(c) IAJAD(MJAD + 1) − IAJAD(1) ≤ LXA

(5) **Error indicator**

IERR value	Meaning	Processing
0	Normal termination.	
3000	Restriction (a) was not satisfied.	Processing is aborted.
3100	Restriction (b) was not satisfied. (Contradictions may exist among input value for N, A, IA, JA).	
3200	Restriction (c) was not satisfied. (Either size of array AJAD or JAJAD is insufficient.)	

(6) **Notes**

(a) When you want to calculate a matrix-vector product of sparse matrix A that has matrices of size 3×3 or 4×4 as block elements, performance will be better if you calculate by using 3.2.25 $\begin{Bmatrix}\text{DAMVJ3}\\\text{RAMVJ3}\end{Bmatrix}$ or 3.2.26 $\begin{Bmatrix}\text{DAMVJ4}\\\text{RAMVJ4}\end{Bmatrix}$.

(b) Matrix storage type conversions preceded by using this subroutine should be executed as less number of times as possible. For example, when you will calculate repeatedly matrix-vector products without changing matrix A for iterative solution methods of simultaneous linear equation, eigenvalue equation of sparse matrix and so on, calculation will be performed efficiently if you perform storage mode conversion only once using 2.2.5 $\begin{Bmatrix}\text{DARSJD}\\\text{RARSJD}\end{Bmatrix}$ or 2.2.6 $\begin{Bmatrix}\text{DARGJM}\\\text{RARGJM}\end{Bmatrix}$ outside the iteration loop and use repeatedly this subroutine inside the iteration loop.

(7) **Example**

(a) Problem
Hold the real random sparse matrix A in the array A as real one-dimensional row-oriented block list type, convert the storage mode into JAD storage type, and then solve the matrix-vector conducts $\boldsymbol{y} = \beta\boldsymbol{y} + \alpha A\boldsymbol{x}$ using by the array AJAD holding the matrix with converted mode.

(b) Main program

```
      PROGRAM BAMVJ1
! *** EXAMPLE OF DAMVJ1 ***
      IMPLICIT NONE
      INTEGER N,M,NZ,ISW,LXA,LXIA
      PARAMETER( N=4, M=1, NZ=8, ISW=0, LXA=NZ, LXIA=N+1 )
      INTEGER IA(N+1),JA(NZ),MJAD,IAJAD(LXIA),JAJAD(LXA),JADORD(N)
      INTEGER IW(N*2+1),IERR
      INTEGER I,J
      REAL(8) A(NZ),AJAD(LXA),X(N),Y(N),W(N)
      REAL(8) ALPHA,BETA
      PARAMETER (ALPHA=1.0D0, BETA=1.0D0)
      CHARACTER*80 FMT1,FMT2,FMT3,FMT4,FMT5
!
      READ(5,*) (IA(I),I=1,N+1)
      READ(5,*) (JA(I),I=1,NZ)
      DO 100 I=1,N
         DO 110 J=IA(I),IA(I+1)-1
            A(J) = J
  110    CONTINUE
  100 CONTINUE
      DO 120 I=1,N
         X(I) = I
         Y(I) = N - I + 1
  120 CONTINUE
!
      WRITE(6,6000)
      WRITE(6,6010) 'IA IN CSR'
      WRITE(FMT1,7000) N+1
      WRITE(6,FMT1) (IA(I),I=1,N+1)
      WRITE(6,6010) 'JA IN CSR:'
      DO 130 I=1,N
         WRITE(FMT2,7000) IA(I+1) - IA(I)
         WRITE(6,FMT2) (JA(J),J=IA(I),IA(I+1)-1)
  130 CONTINUE
      WRITE(6,6010) 'A IN CSR:'
      DO 140 I=1,N
         WRITE(FMT3,7010) IA(I+1) - IA(I)
         WRITE(6,FMT3)( A(J), J=IA(I),IA(I+1)-1 )
  140 CONTINUE
!
      WRITE(FMT4,7010) 1
      WRITE(6,6010) 'ALPHA:'
      WRITE(6,FMT4) ALPHA
      WRITE(6,6010) 'BETA:'
      WRITE(6,FMT4) BETA
      WRITE(6,6010) 'VECTOR X:'
      WRITE(FMT5,7010) N
      WRITE(6,FMT5) (X(I),I=1,N)
      WRITE(6,6010) 'VECTOR Y:'
      WRITE(6,FMT5) (Y(I),I=1,N)
!
!     CONVERT FROM CSR TO JAD
!
      CALL DARGJM&
      (N,M,A,IA,JA,ISW,LXA,LXIA,MJAD,AJAD,IAJAD,JAJAD,JADORD,IW,IERR)
!
!     MATRIX-VECTOR PRODUCT  Y=BETA*Y+ALPHA*A*X
!
      CALL DAMVJ1&
      (AJAD,LXA,IAJAD,JAJAD,JADORD,N,MJAD,ALPHA,BETA,X,Y,W,IERR)
!
      WRITE(6,6030) IERR
      IF( IERR .GE. 3000 ) STOP
!
      WRITE(6,6040)
      WRITE(6,FMT5) (Y(I),I=1,N)
      STOP
!
 6000 FORMAT(/,&
             1X,'***  DAMVJ1  ***',/,/,&
             1X,' **  INPUT  **',/,/,&
             1X,'   * MATRIX DATA FOR CSR FORMAT *')
 6010 FORMAT(/,&
             1X,'     ',A)
 6020 FORMAT(/,&
             1X,'   * ORIGINAL MATRIX *',/)
 6030 FORMAT(/,&
             1X,' **  OUTPUT  **',/,/,&
             1X,'     IERR = ',I4,/)
 6040 FORMAT(/,&
             1X,'   * RESULT Y=BETA*Y+ALPHA*A*X *')
 7000 FORMAT('(1X,5X,',I2,'(3X,I2))')
 7010 FORMAT('(1X,7X,',I2,'(1X,F4.0))')
      END
```

(c) Output results

```
***  DAMVJ1  ***

 **  INPUT  **

   * MATRIX DATA FOR CSR FORMAT *
```

```
        IA IN CSR
            1    3    6    7    9

        JA IN CSR:
            1    3
            1    2    3
            3
            3    4

        A IN CSR:
             1.   2.
             3.   4.   5.
             6.
             7.   8.

        ALPHA:
             1.

        BETA:
             1.

        VECTOR X:
             1.   2.   3.   4.

        VECTOR Y:
             4.   3.   2.   1.

    **  OUTPUT  **

        IERR =    0


      * RESULT Y=BETA*Y+ALPHA*A*X *
            11.  29.  20.  54.
```

3.2.25 DAMVJ3, RAMVJ3 Matrix–Vector Product of a Real Random Sparse Matrix (MJAD; Multiple Jagged Diagonals Storage Type: 3×3 Block Matrix) ($\boldsymbol{y} = \beta\boldsymbol{y} + \alpha A\boldsymbol{x}$)

(1) **Function**

Obtain the product ($\boldsymbol{y} = \beta\boldsymbol{y} + \alpha A\boldsymbol{x}$) of real random sparse matrix A (MJAD; multiple jagged diagonals storage type: 3×3 block matrix) and vector $\boldsymbol{x}$.

(2) **Usage**

Double precision:

```
CALL DAMVJ3 (AJAD, LXA, IAJAD, JAJAD, JADORD, NB, MJAD, ALPHA, BETA, X,
             Y, W, IERR)
```

Single precision:

```
CALL RAMVJ3 (AJAD, LXA, IAJAD, JAJAD, JADORD, NB, MJAD, ALPHA, BETA, X,
             Y, W, IERR)
```

(3) **Arguments**

D:Double precision real	Z:Double precision complex	I: { INTEGER(4) as for 32bit Integer INTEGER(8) as for 64bit Integer }
R:Single precision real	C:Single precision complex	

No.	Argument	Type	Size	Input/ Output	Contents
1	AJAD	{D R}	See Contents	Input	Sparse matrix A (MJAD storage type) **Size**: LXA×3×3 (See Appendix B).
2	LXA	I	1	Input	Size allocated for arrays AJAD and JAJAD.
3	IAJAD	I	MJAD+1	Input	Array of indices for sparse matrix A (MJAD storage type) (See Appendix B).
4	JAJAD	I	LXA	Input	Array of indices for sparse matrix A (MJAD storage type) (See Appendix B).
5	JADORD	I	NB	Input	Array of indices for sparse matrix A (MJAD storage type) (See Appendix B).
6	NB	I	1	Input	Number of block rows (or columns) for dividing matrix A into 3×3 block matrix.
7	MJAD	I	1	Input	Number of jagged diagonals for MJAD storage of matrix A.
8	ALPHA	{D R}	1	Input	Multiplier α of $\boldsymbol{y} = \beta\boldsymbol{y} + \alpha A\boldsymbol{x}$.
9	BETA	{D R}	1	Input	Multiplier β of $\boldsymbol{y} = \beta\boldsymbol{y} + \alpha A\boldsymbol{x}$.

No.	Argument	Type	Size	Input/ Output	Contents
10	X	$\begin{Bmatrix} \text{D} \\ \text{R} \end{Bmatrix}$	NB×3	Input	Vector $\boldsymbol{x}$ of $\boldsymbol{y} = \beta\boldsymbol{y} + \alpha A\boldsymbol{x}$.
11	Y	$\begin{Bmatrix} \text{D} \\ \text{R} \end{Bmatrix}$	NB×3	Input/ Output	Vector $\boldsymbol{y}$ of $\boldsymbol{y} = \beta\boldsymbol{y} + \alpha A\boldsymbol{x}$.
12	W	$\begin{Bmatrix} \text{D} \\ \text{R} \end{Bmatrix}$	NB×3	Work	Work area
13	IERR	I	1	Output	Error indicator

(4) **Restrictions**

(a) NB > 0

(b) $0 <$ MJAD $\leq$ NB

(c) IAJAD(MJAD $+ 1) -$ IAJAD(1) $\leq$ LXA

(5) **Error indicator**

IERR value	Meaning	Processing
0	Normal termination.	
3000	Restriction (a) was not satisfied.	Processing is aborted.
3100	Restriction (b) was not satisfied. (Contradictions may exist among input value for NB, A, IA, JA.)	
3200	Restriction (c) was not satisfied. (Either size of array AJAD or JAJAD is insufficient.)	

(6) **Notes**

(a) This subroutine calculates the matrix-vector product of sparse matrix that have element of 3×3 block matrix. If sparse matrix that have element of 1×1 or 4×4 block matrix, you should be used 3.2.24 $\begin{Bmatrix} \text{DAMVJ1} \\ \text{RAMVJ1} \end{Bmatrix}$ and 3.2.26 $\begin{Bmatrix} \text{DAMVJ4} \\ \text{RAMVJ4} \end{Bmatrix}$, respectively.

(b) Matrix storage type conversions preceded by using this subroutine should be executed as less number of times as possible. For example, when you will calculate repeatedly matrix-vector products without changing matrix A for iterative solution methods of simultaneous linear equation, eigenvalue equation of sparse matrix and so on, calculation will be performed efficiently if you perform storage mode conversion only once using 2.2.5 $\begin{Bmatrix} \text{DARSJD} \\ \text{RARSJD} \end{Bmatrix}$ or 2.2.6 $\begin{Bmatrix} \text{DARGJM} \\ \text{RARGJM} \end{Bmatrix}$ outside the iteration loop and use repeatedly this subroutine inside the iteration loop.

(7) **Example**

(a) Problem

Hold the real random sparse matrix A that have element of 3×3 block matrix in the array A as real one-dimensional row-oriented block list type, convert the storage mode into MJAD storage type, and then solve the matrix-vector conducts $\boldsymbol{y} = \beta\boldsymbol{y} + \alpha A\boldsymbol{x}$ using by the array AJAD holding the matrix with converted mode.

(b) Main program

```
      PROGRAM BAMVJ3
! *** EXAMPLE OF DAMVJ3 ***
      IMPLICIT NONE
      INTEGER NB,M,NZ,ISW,LXA,LXIA,MM
      PARAMETER( NB=4, M=3, NZ=8, ISW=0, LXA=NZ, LXIA=NB+1, MM=M*M )
      INTEGER IA(NB+1),JA(NZ),MJAD,IAJAD(LXIA),JAJAD(LXA),JADORD(NB)
      INTEGER IW(NB*2+1),IERR
      INTEGER I,J,K,L
      REAL(8) A(NZ*MM),AJAD(LXA,MM),X(NB*M),Y(NB*M),W(NB*M)
      REAL(8) ALPHA,BETA
      PARAMETER (ALPHA=1.0D0, BETA=1.0D0)
      CHARACTER*80 FMT1,FMT2,FMT3,FMT4,FMT5
!
      READ(5,*) (IA(I),I=1,NB+1)
      READ(5,*) (JA(I),I=1,NZ)
      DO 100 I=1,NB
         DO 110 J=IA(I),IA(I+1)-1
            DO 120 K=1,MM
               A( (J-1)*MM + K ) = K
  120       CONTINUE
  110    CONTINUE
  100 CONTINUE
      DO 130 I=1,NB
         DO 140 J=1,M
            X( (I-1)*M+J ) = I
            Y( (I-1)*M+J ) = NB - I + 1
  140    CONTINUE
  130 CONTINUE
!
      WRITE(6,6000)
      WRITE(6,6010) 'IA IN BLOCK CSR'
      WRITE(FMT1,7000) NB+1
      WRITE(6,FMT1) (IA(I),I=1,NB+1)
      WRITE(6,6010) 'JA IN BLOCK CSR:'
      DO 150 I=1,NB
         WRITE(FMT2,7000) IA(I+1) - IA(I)
         WRITE(6,FMT2) (JA(J),J=IA(I),IA(I+1)-1)
  150 CONTINUE
      WRITE(6,6010) 'A IN BLOCK CSR:'
      DO 160 I=1,NB
         WRITE(FMT3,7010) (IA(I+1) - IA(I))*M
         DO 170 K=1,M
            WRITE(6,FMT3)&
            ( (A((J-1)*MM+(K-1)*M+L), L=1,M) ,J=IA(I),IA(I+1)-1 )
  170    CONTINUE
  160 CONTINUE
!
      WRITE(FMT4,7010) 1
      WRITE(6,6010) 'ALPHA:'
      WRITE(6,FMT4) ALPHA
      WRITE(6,6010) 'BETA:'
      WRITE(6,FMT4) BETA
      WRITE(6,6010) 'BLOCK VECTOR X:'
      WRITE(FMT5,7010) NB*M
      WRITE(6,FMT5) (X(I),I=1,NB*M)
      WRITE(6,6010) 'BLOCK VECTOR Y:'
      WRITE(6,FMT5) (Y(I),I=1,NB*M)
!
!     CONVERT FROM BLOCK CSR TO JAD
!
      CALL DARGJM&
      (NB,M,A,IA,JA,ISW,LXA,LXIA,MJAD,AJAD,IAJAD,JAJAD,JADORD,IW,IERR)
!
!     MATRIX-VECTOR PRODUCT  Y=BETA*Y+ALPHA*A*X
!
      CALL DAMVJ3&
      (AJAD,LXA,IAJAD,JAJAD,JADORD,NB,MJAD,ALPHA,BETA,X,Y,W,IERR)
!
      WRITE(6,6030) IERR
      IF( IERR .GE. 3000 ) STOP
!
      WRITE(6,6040)
      WRITE(6,6010) 'BLOCK VECTOR Y:'
      WRITE(6,FMT5) (Y(I),I=1,NB*M)
      STOP
!
 6000 FORMAT(/,&
            1X,'***  DAMVJ3  ***',/,/,&
            1X,' **  INPUT  **',/,/,&
            1X,'   * MATRIX DATA FOR CSR FORMAT *')
6010 FORMAT(/,&
            1X,'     ',A)
6020 FORMAT(/,&
            1X,'   * ORIGINAL MATRIX *',/)
6030 FORMAT(/,&
            1X,' **  OUTPUT  **',/,/,&
            1X,'     IERR = ',I4,/)
6040 FORMAT(/,&
            1X,'   * RESULT Y=BETA*Y+ALPHA*A*X *')
7000 FORMAT('(1X,5X,',I2,'(3X,I2))')
7010 FORMAT('(1X,7X,',I2,'(1X,F4.0))')
     END
```

(c) Output results

```
***  DAMVJ3  ***

 **  INPUT  **

   * MATRIX DATA FOR CSR FORMAT *

     IA IN BLOCK CSR
         1    3    6    7    9

     JA IN BLOCK CSR:
         1    3
         1    2    3
         3
         3    4

     A IN BLOCK CSR:
         1.   2.   3.   1.   2.   3.
         4.   5.   6.   4.   5.   6.
         7.   8.   9.   7.   8.   9.
         1.   2.   3.   1.   2.   3.   1.   2.   3.
         4.   5.   6.   4.   5.   6.   4.   5.   6.
         7.   8.   9.   7.   8.   9.   7.   8.   9.
         1.   2.   3.
         4.   5.   6.
         7.   8.   9.
         1.   2.   3.   1.   2.   3.
         4.   5.   6.   4.   5.   6.
         7.   8.   9.   7.   8.   9.

     ALPHA:
          1.

     BETA:
          1.

     BLOCK VECTOR X:
          1.   1.   1.   2.   2.   2.   3.   3.   3.   4.   4.   4.

     BLOCK VECTOR Y:
          4.   4.   4.   3.   3.   3.   2.   2.   2.   1.   1.   1.

 **  OUTPUT  **

     IERR =    0


   * RESULT Y=BETA*Y+ALPHA*A*X *

     BLOCK VECTOR Y:
         28.  64. 100.  39.  93. 147.  20.  47.  74.  43. 106. 169.
```

3.2.26 DAMVJ4, RAMVJ4
Matrix–Vector Product of a Real Random Sparse Matrix (MJAD; Multiple Jagged Diagonals Storage Type: 4×4 Block Matrix) ($\boldsymbol{y} = \beta\boldsymbol{y} + \alpha A\boldsymbol{x}$)

(1) **Function**

Obtain the product ($\boldsymbol{y} = \beta\boldsymbol{y} + \alpha A\boldsymbol{x}$) of real random sparse matrix A (MJAD; multiple jagged diagonals storage type: 4×4 block matrix) and vector $\boldsymbol{y}$.

(2) **Usage**

Double precision:

```
CALL DAMVJ4 (AJAD, LXA, IAJAD, JAJAD, JADORD, NB, MJAD, ALPHA, BETA, X,
             Y, W, IERR)
```

Single precision:

```
CALL RAMVJ4 (AJAD, LXA, IAJAD, JAJAD, JADORD, NB, MJAD, ALPHA, BETA, X,
             Y, W, IERR)
```

(3) **Arguments**

D:Double precision real Z:Double precision complex
R:Single precision real C:Single precision complex
I: {INTEGER(4) as for 32bit Integer, INTEGER(8) as for 64bit Integer}

No.	Argument	Type	Size	Input/ Output	Contents
1	AJAD	{D R}	See Contents	Input	Sparse matrix A (MJAD storage type) **Size**: LXA×4×4 (See Appendix B).
2	LXA	I	1	Input	Size allocated for arrays AJAD and JAJAD.
3	IAJAD	I	MJAD+1	Input	Array of indices for sparse matrix A (MJAD storage type) (See Appendix B).
4	JAJAD	I	LXA	Input	Array of indices for sparse matrix A (MJAD storage type) (See Appendix B).
5	JADORD	I	NB	Input	Array of indices for sparse matrix A (MJAD storage type) (See Appendix B).
6	NB	I	1	Input	Number of block rows (or columns) for dividing matrix A into 4×4 block matrix.
7	MJAD	I	1	Input	Number of jagged diagonals for MJAD storage of matrix A.
8	ALPHA	{D R}	1	Input	Multiplier β of $\boldsymbol{y} = \beta\boldsymbol{y} + \alpha A\boldsymbol{x}$.
9	BETA	{D R}	1	Input	Multiplier α of $\boldsymbol{y} = \beta\boldsymbol{y} + \alpha A\boldsymbol{x}$.

No.	Argument	Type	Size	Input/ Output	Contents
10	X	$\begin{Bmatrix}\text{D}\\\text{R}\end{Bmatrix}$	NB×4	Input	Vector $\boldsymbol{x}$ of $\boldsymbol{y} = \beta\boldsymbol{y} + \alpha A\boldsymbol{x}$.
11	Y	$\begin{Bmatrix}\text{D}\\\text{R}\end{Bmatrix}$	NB×4	Input/ Output	Vector $\boldsymbol{y}$ of $\boldsymbol{y} = \beta\boldsymbol{y} + \alpha A\boldsymbol{x}$.
12	W	$\begin{Bmatrix}\text{D}\\\text{R}\end{Bmatrix}$	NB×4	Work	Work area
13	IERR	I	1	Output	Error indicator

(4) **Restrictions**

(a) $\text{NB} > 0$

(b) $0 < \text{MJAD} \leq \text{NB}$

(c) $\text{IAJAD}(\text{MJAD} + 1) - \text{IAJAD}(1) \leq \text{LXA}$

(5) **Error indicator**

IERR value	Meaning	Processing
0	Normal termination.	
3000	Restriction (a) was not satisfied.	Processing is aborted.
3100	Restriction (b) was not satisfied. (Contradictions may exist among input value for NB, A, IA, JA.)	
3200	Restriction (c) was not satisfied. (Either size of array AJAD or JAJAD is insufficient.)	

(6) **Notes**

(a) This subroutine calculates the matrix-vector product of sparse matrix that have element of 4×4 block matrix. If sparse matrix that have element of 1×1 or 3×3 block matrix, you should be used 3.2.24 $\begin{Bmatrix}\text{DAMVJ1}\\\text{RAMVJ1}\end{Bmatrix}$ and 3.2.25 $\begin{Bmatrix}\text{DAMVJ3}\\\text{RAMVJ3}\end{Bmatrix}$, respectively.

(b) Matrix storage type conversions preceded by using this subroutine should be executed as less number of times as possible. For example, when you will calculate repeatedly matrix-vector products without changing matrix A for iterative solution methods of simultaneous linear equation, eigenvalue equation of sparse matrix and so on, calculation will be performed efficiently if you perform storage mode conversion only once using 2.2.5 $\begin{Bmatrix}\text{DARSJD}\\\text{RARSJD}\end{Bmatrix}$ or 2.2.6 $\begin{Bmatrix}\text{DARGJM}\\\text{RARGJM}\end{Bmatrix}$ outside the iteration loop and use repeatedly this subroutine inside the iteration loop.

(7) **Example**

(a) Problem

Hold the real random sparse matrix A that have element of 4×4 block matrix in the array A as real one-dimensional row-oriented block list type, convert the storage mode into MJAD storage type, and then solve the matrix-vector conducts $\boldsymbol{y} = \beta\boldsymbol{y} + \alpha A\boldsymbol{x}$ using by the array AJAD holding the matrix with converted mode.

(b) Main program

```
      PROGRAM BAMVJ4
! *** EXAMPLE OF DAMVJ4 ***
      IMPLICIT NONE
      INTEGER NB,M,NZ,ISW,LXA,LXIA,MM
      PARAMETER( NB=4, M=4, NZ=8, ISW=0, LXA=NZ, LXIA=NB+1, MM=M*M )
      INTEGER IA(NB+1),JA(NZ),MJAD,IAJAD(LXIA),JAJAD(LXA),JADORD(NB)
      INTEGER IW(NB*2+1),IERR
      INTEGER I,J,K,L
      REAL(8) A(NZ*MM),AJAD(LXA,MM),X(NB*M),Y(NB*M),W(NB*M)
      REAL(8) ALPHA,BETA
      PARAMETER (ALPHA=1.0D0, BETA=1.0D0)
      CHARACTER*80 FMT1,FMT2,FMT3,FMT4,FMT5
!
      READ(5,*) (IA(I),I=1,NB+1)
      READ(5,*) (JA(I),I=1,NZ)
      DO 100 I=1,NB
         DO 110 J=IA(I),IA(I+1)-1
            DO 120 K=1,MM
               A( (J-1)*MM + K ) = K
  120       CONTINUE
  110    CONTINUE
  100 CONTINUE
      DO 130 I=1,NB
         DO 140 J=1,M
            X( (I-1)*M+J ) = I
            Y( (I-1)*M+J ) = NB - I + 1
  140    CONTINUE
  130 CONTINUE
!
      WRITE(6,6000)
      WRITE(6,6010) 'IA IN BLOCK CSR'
      WRITE(FMT1,7000) NB+1
      WRITE(6,FMT1) (IA(I),I=1,NB+1)
      WRITE(6,6010) 'JA IN BLOCK CSR:'
      DO 150 I=1,NB
         WRITE(FMT2,7000) IA(I+1) - IA(I)
         WRITE(6,FMT2) (JA(J),J=IA(I),IA(I+1)-1)
  150 CONTINUE
      WRITE(6,6010) 'A IN BLOCK CSR:'
      DO 160 I=1,NB
         WRITE(FMT3,7010) (IA(I+1) - IA(I))*M
         DO 170 K=1,M
            WRITE(6,FMT3)&
            ( (A((J-1)*MM+(K-1)*M+L), L=1,M) ,J=IA(I),IA(I+1)-1 )
  170    CONTINUE
  160 CONTINUE
!
      WRITE(FMT4,7010) 1
      WRITE(6,6010) 'ALPHA:'
      WRITE(6,FMT4) ALPHA
      WRITE(6,6010) 'BETA:'
      WRITE(6,FMT4) BETA
      WRITE(6,6010) 'BLOCK VECTOR X:'
      WRITE(FMT5,7010) NB*M
      WRITE(6,FMT5) (X(I),I=1,NB*M)
      WRITE(6,6010) 'BLOCK VECTOR Y:'
      WRITE(6,FMT5) (Y(I),I=1,NB*M)
!
!     CONVERT FROM BLOCK CSR TO JAD
!
      CALL DARGJM&
      (NB,M,A,IA,JA,ISW,LXA,LXIA,MJAD,AJAD,IAJAD,JAJAD,JADORD,IW,IERR)
!
!     MATRIX-VECTOR PRODUCT  Y=BETA*Y+ALPHA*A*X
!
      CALL DAMVJ4&
      (AJAD,LXA,IAJAD,JAJAD,JADORD,NB,MJAD,ALPHA,BETA,X,Y,W,IERR)
!
      WRITE(6,6030) IERR
      IF( IERR .GE. 3000 ) STOP
!
      WRITE(6,6040)
      WRITE(6,6010) 'BLOCK VECTOR Y:'
      WRITE(6,FMT5) (Y(I),I=1,NB*M)
      STOP
!
 6000 FORMAT(/,&
            1X,'***  DAMVJ4  ***',/,/,&
            1X,' **  INPUT  **',/,/,&
            1X,'   * MATRIX DATA FOR CSR FORMAT *')
6010 FORMAT(/,&
            1X,'     ',A)
6020 FORMAT(/,&
            1X,'   * ORIGINAL MATRIX *',/)
6030 FORMAT(/,&
            1X,' **  OUTPUT  **',/,/,&
            1X,'     IERR = ',I4,/)
6040 FORMAT(/,&
            1X,'   * RESULT Y=BETA*Y+ALPHA*A*X *')
7000 FORMAT('(1X,5X,',I2,'(3X,I2))')
7010 FORMAT('(1X,7X,',I2,'(1X,F4.0))')
     END
```

(c) Output results

```
***  DAMVJ4  ***

 **  INPUT  **

   * MATRIX DATA FOR CSR FORMAT *

     IA IN BLOCK CSR
         1    3    6    7    9

     JA IN BLOCK CSR:
         1    3
         1    2    3
         3
         3    4

     A IN BLOCK CSR:
         1.   2.   3.   4.   1.   2.   3.   4.
         5.   6.   7.   8.   5.   6.   7.   8.
         9.  10.  11.  12.   9.  10.  11.  12.
        13.  14.  15.  16.  13.  14.  15.  16.
         1.   2.   3.   4.   1.   2.   3.   4.   1.   2.   3.   4.
         5.   6.   7.   8.   5.   6.   7.   8.   5.   6.   7.   8.
         9.  10.  11.  12.   9.  10.  11.  12.   9.  10.  11.  12.
        13.  14.  15.  16.  13.  14.  15.  16.  13.  14.  15.  16.
         1.   2.   3.   4.
         5.   6.   7.   8.
         9.  10.  11.  12.
        13.  14.  15.  16.
         1.   2.   3.   4.   1.   2.   3.   4.
         5.   6.   7.   8.   5.   6.   7.   8.
         9.  10.  11.  12.   9.  10.  11.  12.
        13.  14.  15.  16.  13.  14.  15.  16.

     ALPHA:
          1.

     BETA:
          1.

     BLOCK VECTOR X:
          1.   1.   1.   1.   2.   2.   2.   2.   3.   3.   3.   3.   4.   4.   4.   4.

     BLOCK VECTOR Y:
          4.   4.   4.   4.   3.   3.   3.   3.   2.   2.   2.   2.   1.   1.   1.   1.

 **  OUTPUT  **

     IERR =    0


   * RESULT Y=BETA*Y+ALPHA*A*X *

     BLOCK VECTOR Y:
         44. 108. 172. 236.  63. 159. 255. 351.  32.  80. 128. 176.  71. 183. 295. 407.
```

3.2.27 ZANVJ1, CANVJ1 Matrix–Vector Product of a Complex Random Sparse Matrix (JAD; Jagged Diagonals Storage Type) ($\boldsymbol{y} = \beta \boldsymbol{y} + \alpha A \boldsymbol{x}$)

(1) **Function**

Obtain the product ($\boldsymbol{y} = \beta \boldsymbol{y} + \alpha A \boldsymbol{x}$) of complex random sparse matrix A (JAD; jagged diagonals storage type) and vector $\boldsymbol{x}$.

(2) **Usage**

Double precision:

CALL ZANVJ1 (AJAD, LXA, IAJAD, JAJAD, JADORD, N, MJAD, ALPHA, BETA, X, Y, W, IERR)

Single precision:

CALL CANVJ1 (AJAD, LXA, IAJAD, JAJAD, JADORD, N, MJAD, ALPHA, BETA, X, Y, W, IERR)

(3) **Arguments**

D:Double precision real Z:Double precision complex I: $\left\{\begin{array}{l}\text{INTEGER(4) as for 32bit Integer}\\ \text{INTEGER(8) as for 64bit Integer}\end{array}\right\}$
R:Single precision real C:Single precision complex

No.	Argument	Type	Size	Input/ Output	Contents
1	AJAD	$\left\{\begin{matrix}Z\\C\end{matrix}\right\}$	LXA	Input	Sparse matrix A (JAD storage type) (See Appendix B).
2	LXA	I	1	Input	Size allocated for arrays AJAD and JAJAD.
3	IAJAD	I	MJAD+1	Input	Array of indices for sparse matrix A (JAD storage type) (See Appendix B).
4	JAJAD	I	LXA	Input	Array of indices for sparse matrix A (JAD storage type) (See Appendix B).
5	JADORD	I	N	Input	Array of indices for sparse matrix A (JAD storage type) (See Appendix B).
6	N	I	1	Input	Order of vectors X and Y.
7	MJAD	I	1	Input	Number of jagged diagonals for JAD storage of matrix A.
8	ALPHA	$\left\{\begin{matrix}Z\\C\end{matrix}\right\}$	1	Input	Multiplier α of $\boldsymbol{y} = \beta \boldsymbol{y} + \alpha A \boldsymbol{x}$.
9	BETA	$\left\{\begin{matrix}Z\\C\end{matrix}\right\}$	1	Input	Multiplier β of $\boldsymbol{y} = \beta \boldsymbol{y} + \alpha A \boldsymbol{x}$.
10	X	$\left\{\begin{matrix}Z\\C\end{matrix}\right\}$	N	Input	Vector $\boldsymbol{x}$ of $\boldsymbol{y} = \beta \boldsymbol{y} + \alpha A \boldsymbol{x}$.
11	Y	$\left\{\begin{matrix}Z\\C\end{matrix}\right\}$	N	Input/ Output	Vector $\boldsymbol{y}$ of $\boldsymbol{y} = \beta \boldsymbol{y} + \alpha A \boldsymbol{x}$.

No.	Argument	Type	Size	Input/ Output	Contents
12	W	$\begin{Bmatrix} \text{Z} \\ \text{C} \end{Bmatrix}$	N	Work	Work area
13	IERR	I	1	Output	Error indicator

(4) **Restrictions**

(a) N > 0

(b) 0 < MJAD ≤ N

(c) IAJAD(MJAD + 1) − IAJAD(1) ≤ LXA

(5) **Error indicator**

IERR value	Meaning	Processing
0	Normal termination.	
3000	Restriction (a) was not satisfied.	Processing is aborted.
3100	Restriction (b) was not satisfied. (Contradictions may exist among input value for N, A, IA, JA.)	
3200	Restriction (c) was not satisfied. (Either size of array AJAD or JAJAD is insufficient.)	

(6) **Notes**

(a) Matrix storage type conversions preceded by using this subroutine should be executed as less number of times as possible. For example, when you will calculate repeatedly matrix-vector products without changing matrix A for iterative solution methods of simultaneous linear equation, eigenvalue equation of sparse matrix and so on, calculation will be performed efficiently if you perform storage mode conversion only once using 2.2.7 $\begin{Bmatrix} \text{ZARSJD} \\ \text{CARSJD} \end{Bmatrix}$ or 2.2.8 $\begin{Bmatrix} \text{ZARGJM} \\ \text{CARGJM} \end{Bmatrix}$ outside the iteration loop and use repeatedly this subroutine inside the iteration loop.

(7) **Example**

(a) Problem
Hold the complex random sparse matrix A in the array A as real one-dimensional row-oriented block list type, convert the storage mode into JAD storage type, and then solve the matrix-vector conducts $\boldsymbol{y} = \beta\boldsymbol{y} + \alpha A\boldsymbol{x}$ using by the array AJAD holding the matrix with converted mode.

(b) Main program

```
      PROGRAM AANVJ1
! *** EXAMPLE OF AANVJ1 ***
      IMPLICIT NONE
      INTEGER N,M,NZ,ISW,LXA,LXIA
      PARAMETER( N=4, M=1, NZ=8, ISW=0, LXA=NZ, LXIA=N+1 )
      INTEGER IA(N+1),JA(NZ),MJAD,IAJAD(LXIA),JAJAD(LXA),JADORD(N)
      INTEGER IW(N*2+1),IERR
      INTEGER I,J
      COMPLEX(8) A(NZ),AJAD(LXA),X(N),Y(N),W(N)
      COMPLEX(8) ALPHA,BETA
      PARAMETER (ALPHA=(1.0D0,0.D0), BETA=(1.0D0,0.D0))
      CHARACTER*80 FMT1,FMT2,FMT3,FMT4,FMT5,FMT6
!
      READ(5,*) (IA(I),I=1,N+1)
      READ(5,*) (JA(I),I=1,NZ)
      DO 100 I=1,N
         DO 110 J=IA(I),IA(I+1)-1
            A(J) = CMPLX(J,-J, KIND=8)
  110    CONTINUE
  100 CONTINUE
      DO 120 I=1,N
         X(I) = CMPLX(I,-I, KIND=8)
         Y(I) = CMPLX(N - I + 1, -N + I - 1, KIND=8)
  120 CONTINUE
!
      WRITE(6,6000)
      WRITE(6,6010) 'IA IN CSR'
      WRITE(FMT1,7000) N+1
      WRITE(6,FMT1) (IA(I),I=1,N+1)
      WRITE(6,6010) 'JA IN CSR:'
      DO 130 I=1,N
         WRITE(FMT2,7000) IA(I+1) - IA(I)
         WRITE(6,FMT2) (JA(J),J=IA(I),IA(I+1)-1)
  130 CONTINUE
      WRITE(6,6010) 'A IN CSR:'
      DO 140 I=1,N
         WRITE(FMT3,7010) IA(I+1) - IA(I)
         WRITE(6,FMT3)( A(J), J=IA(I),IA(I+1)-1 )
  140 CONTINUE
!
      WRITE(FMT4,7010) 1
      WRITE(6,6010) 'ALPHA:'
      WRITE(6,FMT4) ALPHA
      WRITE(6,6010) 'BETA:'
      WRITE(6,FMT4) BETA
      WRITE(6,6010) 'VECTOR X:'
      WRITE(FMT5,7010) N
      WRITE(6,FMT5) (X(I),I=1,N)
      WRITE(6,6010) 'VECTOR Y:'
      WRITE(6,FMT5) (Y(I),I=1,N)
!
!     CONVERT FROM CSR TO JAD
!
      CALL ZARGJM&
      (N,M,A,IA,JA,ISW,LXA,LXIA,MJAD,AJAD,IAJAD,JAJAD,JADORD,IW,IERR)
!
!     MATRIX-VECTOR PRODUCT  Y=BETA*Y+ALPHA*A*X
!
      CALL ZANVJ1&
      (AJAD,LXA,IAJAD,JAJAD,JADORD,N,MJAD,ALPHA,BETA,X,Y,W,IERR)
!
      WRITE(6,6030) IERR
      IF( IERR .GE. 3000 ) STOP
!
      WRITE(6,6040)
      WRITE(FMT6,7020) N
      WRITE(6,FMT6) (Y(I),I=1,N)
      STOP
!
 6000 FORMAT(/,&
             1X,'***  ZANVJ1  ***',/,/,&
             1X,' **  INPUT  **',/,/,&
             1X,'   * MATRIX DATA FOR CSR FORMAT *')
 6010 FORMAT(/,&
             1X,'     ',A)
 6020 FORMAT(/,&
             1X,'   * ORIGINAL MATRIX *',/)
 6030 FORMAT(/,&
             1X,' **  OUTPUT  **',/,/,&
             1X,'     IERR = ',I4,/)
 6040 FORMAT(/,&
             1X,'   * RESULT Y=BETA*Y+ALPHA*A*X *')
 7000 FORMAT('(1X,5X,',I2,'(3X,I2))')
 7010 FORMAT('(1X,7X,',I2,'(" (",F4.1,",",F4.1,")"))')
 7020 FORMAT('(1X,7X,',I2,'(" (",F4.1,",",F6.1,")"))')
      END
```

(c) Output results

```
***  ZANVJ1  ***

 **  INPUT  **

   * MATRIX DATA FOR CSR FORMAT *

     IA IN CSR
         1    3    6    7    9

     JA IN CSR:
         1    3
         1    2    3
         3
         3    4

     A IN CSR:
        ( 1.0,-1.0) ( 2.0,-2.0)
        ( 3.0,-3.0) ( 4.0,-4.0) ( 5.0,-5.0)
        ( 6.0,-6.0)
        ( 7.0,-7.0) ( 8.0,-8.0)

     ALPHA:
        ( 1.0, 0.0)

     BETA:
        ( 1.0, 0.0)

     VECTOR X:
        ( 1.0,-1.0) ( 2.0,-2.0) ( 3.0,-3.0) ( 4.0,-4.0)

     VECTOR Y:
        ( 4.0,-4.0) ( 3.0,-3.0) ( 2.0,-2.0) ( 1.0,-1.0)

 **  OUTPUT  **

     IERR =    0


   * RESULT Y=BETA*Y+ALPHA*A*X *
        ( 4.0, -18.0) ( 3.0, -55.0) ( 2.0, -38.0) ( 1.0,-107.0)
```

Chapter 4

EIGENVALUES AND EIGENVECTORS

4.1 INTRODUCTION

This chapter describes subroutines that obtain eigenvalues and eigenvectors of matrices.

In standard eigenvalue problem, obtain the value λ and corresponding vector $\boldsymbol{x}$ which satisfy the following equation for a given matrix A:

$A\boldsymbol{x} = \lambda\boldsymbol{x}$.

The value λ and the corresponding vector $\boldsymbol{x}$ are called an eigenvalue and the corresponding eigenvector, respectively. In generalized eigenvalue problem, obtain the value λ and corresponding vector $\boldsymbol{x}$ which satisfy one of the following equations for given matrices A and B:

$A\boldsymbol{x} = \lambda B\boldsymbol{x}$

or

$AB\boldsymbol{x} = \lambda\boldsymbol{x}$ (both A and B are Hermitian, B is positive definite)

or

$BA\boldsymbol{x} = \lambda\boldsymbol{x}$ (both A and B are Hermitian, B is positive definite).

These λ and $\boldsymbol{x}$ are also called an eigenvalue and an eigenvector. Various procedures have been devised to solve the eigenvalue problem depending on the type of matrix. If both A and B are Hermitian and B is positive definite, all the eigenvalues are real and the eigenvectors for different eigenvalues are orthogonal for each other.

This library uses the following three-step process for solving the eigenvalue problem as a rule.

(1) Transform the input matrix to a real symmetric tridiagonal matrix or Hessenberg matrix.

(2) Obtain the eigenvalues and eigenvectors of the real symmetric tridiagonal or Hessenberg matrix.

(3) Transform the obtained eigenvectors to the eigenvectors of the original input matrix.

The subroutines contained in this chapter provide functions corresponding to the following six categories.

All eigenvalues and all eigenvectors: Obtain all eigenvalues and the corresponding eigenvectors.

All eigenvalues: Obtain all eigenvalues only.

Eigenvalues and eigenvectors: Obtain a number of the largest or smallest eigenvalues and the corresponding eigenvectors.

Eigenvalues: Obtain a number of the largest or smallest eigenvalues in a specified interval.

Eigenvalues and eigenvectors (Interval Specified): Obtain a number of the largest or smallest eigenvalues in a specified interval and the corresponding eigenvectors.

Eigenvalues (Interval Specified): Obtain a number of the largest or smallest eigenvalues.

However, only functions corresponding to ‘all eigenvalues and all eigenvectors’ and ‘all eigenvalues’ are provided for an asymmetric matrix.

4.1.1 Notes

(1) When using these subroutines, you must first select the appropriate subroutine group for the matrix type (for example Section 4.2, “Real Matrix” or Section 4.9, “Real Symmetric Band Matrix”) and then select the optimum subroutine based on your objectives, the processing time, memory requirements, and so on.

(2) In general, functions corresponding to ‘All eigenvalues and all eigenvectors’ or ‘Eigenvalues and eigenvectors’ require more processing time and memory than functions corresponding to ‘All eigenvalues’ or ‘Eigenvalues’ respectively.

(3) In general, it is more efficient to use functions corresponding to ‘Eigenvalues and eigenvectors’ or ‘Eigenvalues’ if you want to obtain no more than 20% of the total number of eigenvalues. To obtain more than 20% of the total number of eigenvalues, functions corresponding to ‘All eigenvalues and all eigenvectors’ or ‘All eigenvalues’ require less processing time.

(4) In this library, the subroutines of the generalized eigenvalue problem require the restriction that B is positive definite. In the following cases, however, the eigenvalues and the eigenvectors can be obtained even if B is not positive definite.

(a) Matrix B is not positive definite but matrix A is positive definite

$$Bv = \lambda^{-1} Av$$

gives non-zero eigenvalues.

(b) Both of A and B are not positive definite but $A + B$ is positive definite

$$Av = \frac{\lambda}{1+\lambda}(A + B)v$$

gives the eigenvalues which are not -1.

(5) If the input matrix is a symmetric matrix or Hermitian matrix, the use of the exclusive subroutines requires less processing time.

(6) Matrix structure

The subroutines for regular sparse matrices are used to solve eigenvalue equations of regular sparse matrices which are produced in the two-dimensional finite difference method or in the three-dimensional implicit-solution finite difference method. To solve eigenvalue equations of random sparse matrices produced in other difference methods or in the finite element method, subroutines for random sparse matrices are used. (See Appendix B for the matrix data storage method.)

4.1.2 Algorithms Used

4.1.2.1 Transforming a real matrix to a Hessenberg matrix

A basic similarity transformation is used to transform an $n \times n$ real matrix A to a Hessenberg matrix $H = (h_{ij}) : h_{ij} = 0 (i > j+1)$. That is,

$$A_1 = A$$

is assumed, and the Hessenberg matrix is obtained by iterating the transformation: $n-2$ times,

$$A_{k+1} = P_{k+1}^{-1} I_{k+1,(k+1)'} A_k I_{k+1,(k+1)'} P_{k+1}$$

where $I_{k+1,(k+1)}$and P_{k+1} are the substitution matrix and similarity transformation matrix respectively determined by (1) and (2) below. A_k has a Hessenberg format for the first $k-1$ columns. If we let:

$$A_k = (a_{ij}^{(k)})$$

then:

(1) From column k, find:

$$|a_{(k+1)',k}^{(k)}| = \max_{i=k+1,\cdots,n} |a_{ik}^{(k)}|$$

and exchange row $(k+1)'$ with row $(k+1)$ and column $(k+1)$ with column $(k+1)'$. If this value is 0, the matrix is decomposed into two submatrices, and you should solve the eigenvalue problem for these two submatrices.

(2)
```
for  $i = k+2, n$
    $p_{i,k+1}^{(k+1)} \leftarrow \frac{a_{ik}^{(k)}}{a_{k+1,k}^{(k)}}$
    Row $i$ ← Row $i$ − Column $(k+1) \times p_{i,k+1}^{(k+1)}$
    Column $(k+1)$ ← Column $(k+1)$ + Row $i \times p_{i,k+1}^{(k+1)}$
```

$$\begin{cases} (P_{k+1})_{i,k+1} = p_{i,k+1}^{(k+1)} & (i = k+2, \cdots, n) \\ (P_{k+1})_{ij} = \delta_{ij} & (\text{For other } i \text{ and } j) \\ & (\delta_{ij}\text{is the Kronecker delta}) \end{cases}$$

Note In general, if you cumulate the transformation matrix, you can obtain the eigenvectors. That is, if you use a non-singular matrix to repeatedly apply the transformation to the $m \times m$ matrix A to obtain the following relationship:

$$Q_m^{-1} \cdots Q_2^{-1} Q_1^{-1} A Q_1 Q_2 \cdots Q_m = \Lambda$$

where Λ is a diagonal matrix, then the eigenvalues become the diagonal components of and each of the column vectors in the product of transformation matrices $(Q_1 Q_2 \cdots Q_m)$ becomes an eigenvector.

4.1.2.2 Transforming a complex matrix to a Hessenberg matrix

The Householder method is used to transform an $n \times n$ complex matrix A to a Hessenberg matrix. That is, $A_1 = A$ is assumed, and for $k = 1, 2, \cdots, n-2$, a vector $\boldsymbol{u}_k$ is taken so that:

$$H_k = \frac{1}{2} \boldsymbol{u}_k^* \boldsymbol{u}_k$$

$$P_k = I - \frac{\boldsymbol{u}_k \boldsymbol{u}_k^*}{H_k}$$

and all elements below the subdiagonal component is column k of:

$$A_{k+1} = P_k A_k P_k$$

become 0. A_{n-1} becomes the obtained Hessenberg matrix. In addition, the transformation matrix P_k is a unitary Hermitian matrix.

4.1.2.3 Balancing real and complex matrices

Real and complex matrices are balanced before they are transformed to Hessenberg matrices. First, the original matrix A is multiplied on the left and right by P in which rows and columns have been suitably exchanged to form:

$$PAP = \begin{bmatrix} U & X & Y \\ O & B & Z \\ O & O & V \end{bmatrix}$$

U and V are upper triangular matrices having self-evident isolated eigenvalues as diagonal components, and B is a square matrix that does not contain further isolated eigenvalues.
Next, $B_1 = B$ is assumed and the non-singular diagonal matrix D_k is used to repeatedly perform the similarity transformation:

$$B_{k+1} = D_k^{-1} B_k D_k$$

unit the absolute sum of mutually corresponding rows and columns of B are nearly equal. Eventually, this is transformed to the following form:

$$\begin{bmatrix} U & XD & Y \\ O & D^{-1}BD & D^{-1}Z \\ O & O & V \end{bmatrix}$$

4.1.2.4 QR method and double QR method

For a non-singular complex Hessenberg matrix H, there is a unitary matrix Q and an upper triangular matrix R (for which all diagonal components are positive) such that H is uniquely decomposed into $H = QR$. $H_1 = H$ is assumed, H_k is decomposed into $H_k = Q_k R_k$, and these are multiplied in the reverse order to form:

$$H_{k+1} = R_k Q_k = Q_k^* H_k Q_k \quad (Q_k^* \text{ is the adjoint matrix of } Q_k)$$

If $H_1, H_2, \cdots, H_{k-1}, H_k$ are created, they are all Hessenberg matrices. As $k \to \infty, H_k$ converges to an upper triangular matrix having the eigenvalues of H as its diagonal components. To accelerate convergence in the actual QR method, $H_k - \mu_k I$ is created in place of H_k by performing a translation of the origin and this is decomposed into:

$$H_k - \mu_k I = Q_k R_k \quad (\text{where } \mu_k \text{ is taken as an approximation of the eigenvalue})$$

If $H_{k+1} = R_k Q_k + \mu_k I$ is created, then H_{k+1} becomes:

$$H_{k+1} = Q_k^* H_k Q_k$$

After iterating this operation until the sequence of matrices converges, the values adjusted by the cumulative amount the origin was translated become the eigenvalues.

Double QR method

If the origin is translated as described above for a real matrix (asymmetric), a complex matrix may appear at an intermediate step. To prevent this, we let:

$$H_{k+2} = (Q_k Q_{k+1})^T H_k (Q_k Q_{k+1})$$

$$(H_k - \mu_k I)(H_k - \mu_{k+1} I) = (Q_k Q_{k+1})(R_{k+1} R_k)$$

If μ_k and μ_{k+1} both become real or conjugate complex, then the left hand side of the second equation shown above becomes a real matrix.

Actually, using the Householder transformation matrix $P_i i$:

$$Q_k Q_{k+1} = P_1 P_2 \cdots\cdots P_{n-1}$$

is assumed, and H_{k+2} becomes:

$$H_{k+2} = P_{n-1}^T \cdots\cdots P_2^T P_1^T H_k P_1 P_2 \cdots\cdots P_{n-1}$$

For details, refer to (1), (2), and (5) in the reference bibliography.

4.1.2.5 Transforming a real symmetric matrix to a real symmetric tridiagonal matrix

The Householder method is used to transform an $n \times n$ real symmetric matrix A to a real symmetric tridiagonal matrix T. That is, $A_1 = A$ is assumed, and for $k = 1, 2, \cdots, n-2$, a vector u_k is taken so that:

$$H_k = \frac{1}{2} \boldsymbol{u}_k^T \boldsymbol{u}_k$$

$$P_k = I - \frac{\boldsymbol{u}_k \boldsymbol{u}_k^T}{H_k}$$

and all elements below the subdiagonal component in column k of:

$$A_{k+1} = P_k A_k P_k$$

become 0. A_{n-1} becomes the obtained real symmetric tridiagonal matrix. In addition, the transformation matrix P_k is an orthogonal symmetric matrix.

4.1.2.6 Transforming a Hermitian matrix to a real symmetric tridiagonal matrix

First, the Householder method is used to transform an $n \times n$ Hermitian matrix A to a Hermitian tridiagonal matrix S.

$$S = P_{n-2} \cdots P_2 P_1 A P_1 P_2 \cdots P_{n-2}$$

Then a regular complex diagonal matrix D is used (similarity transformation) to transform matrix S to a real symmetric tridiagonal matrix T.

$$T = D^* S D$$

4.1.2.7 The Householder transformation by block algorithm

As for the Householder transformation, the block algorithm is used. This method simplifies the update of a matrix by applying the lump sum of a rank-one matrix update that transforms the original real symmetric matrix into a real symmetric tridiagonal matrix. Let A_{k+1} be the symmetric matrix that is generated after similarity transformations are performed for k times on the original symmetric matrix A. Then it holds that:

$$A_{k+1} = P_k A_k P_k = A_k - \boldsymbol{u}_k \boldsymbol{v}_k^T - \boldsymbol{v}_k \boldsymbol{u}_k^T$$

where P_k is an orthogonal matrix. Also the following relationship holds:

$$A_1 = A$$

$$P_k = I - \frac{\boldsymbol{u}_k \boldsymbol{u}_k^T}{H_k}$$

$$H_k = \frac{1}{2}\boldsymbol{u}_k^T \boldsymbol{u}_k$$

$$\boldsymbol{y}_k = A_k \boldsymbol{u}_k$$

$$\boldsymbol{v}_k = \frac{(\boldsymbol{y}_k - \frac{(\boldsymbol{u}_k^T \boldsymbol{y}_k)\boldsymbol{u}_k}{2H_k})}{H_k}$$

The Householder transformation updates the symmetric matrix twice for one similarity transformation.
The A_{k+1} can be expressed without using A_k explicitly.

$$A_{k+1} = A_{k-1} - \boldsymbol{u}_{k-1}\boldsymbol{v}_{k-1}{}^T - \boldsymbol{v}_{k-1}\boldsymbol{u}_{k-1}{}^T - \boldsymbol{u}_k\boldsymbol{v}_k{}^T - \boldsymbol{v}_k\boldsymbol{u}_k{}^T$$

Similarly, matrix A_{p+1} after the p times of mirror transformation becomes:

$$A_{p+1} = A_1 - \sum_{i=1}^{p}(\boldsymbol{u}_i \boldsymbol{v}_i^T + \boldsymbol{v}_i \boldsymbol{u}_i^T)$$

Therefore, matrix A_{p+1} can be computed at a higher speed if matrix A_1 is updated in the lump, once per $2p$ matrices. If the original matrix is Hermitian, the transpose notation "^{T}" should be replaced with the Hermite conjugate notation "*". For details, refer to (3) and (4) in the reference bibliography.

4.1.2.8 Transforming a real symmetric band matrix to a real symmetric tridiagonal matrix

The Givens method is used to transform an $n \times n$ real symmetric band matrix A (band width MB) to a real symmetric tridiagonal matrix T. That is, with a real symmetric band matrix A and the transformation matrix

P assumed that:

$$P = \begin{array}{cc} \begin{array}{ccccccccccc} & & & i\text{-th column} & & & & j\text{-th column} & & & \end{array} & \\ \left[\begin{array}{ccccccccccc} 1 & & & \vdots & & & & \vdots & & & \\ & \ddots & & \vdots & & & & \vdots & & 0 & \\ & & 1 & \vdots & & & & \vdots & & & \\ \cdots & \cdots & \cdots & \cos\theta & \cdots & \cdots & \cdots & \sin\theta & \cdots & \cdots & \cdots \\ & & & \vdots & 1 & & & \vdots & & & \\ & & & \vdots & & \ddots & & \vdots & & & \\ & & & \vdots & & & 1 & \vdots & & & \\ \cdots & \cdots & \cdots & -\sin\theta & \cdots & \cdots & \cdots & \cos\theta & \cdots & \cdots & \cdots \\ & & & \vdots & & & & \vdots & 1 & & \\ & 0 & & \vdots & & & & \vdots & & \ddots & \\ & & & \vdots & & & & \vdots & & & 1 \end{array}\right] & \begin{array}{l} \\ \\ \\ i\text{-th row} \\ \\ \\ \\ j\text{-th row} \\ \\ \\ \\ \end{array} \end{array}$$

where

$$\cos\theta = \frac{a_{j,j-1}}{\sqrt{a_{j,j+1}^2 + a_{i,j-1}^2}}$$

$$\sin\theta = \frac{a_{i,j-1}}{\sqrt{a_{j,j-1}^2 + a_{i,j-1}^2}},$$

$a_{i,j-1}$ can be 0 by transforming $A' = P^T AP$ ($a_{i,j}$ is the (i, j) component of matrix A).
If this transformation is used for

$$j = 2, \cdots\cdots\cdots\cdots, n-1$$

$$i = j+1, \cdots\cdots, \text{MIN}(MB + j - 1, n)$$

A real symmetric band matrix A can be a real Symmetric tridiagonal matrix T. At this time the number of transformation is $(MB - 1)(n - \frac{MB}{2} - 1)$. The transformation matrix P is an orthogonal matrix.

4.1.2.9 QR method

For a tridiagonal matrix T, there is a unitary matrix Q and an upper triangular matrix R (for which all diagonal components are positive) such that T is uniquely decomposed into $T = QR$. $T_k = T$ is assumed, T_k is decomposed into $T_k = Q_k R_k$, and these are multiplied in the reverse order to form:

$$T_{k+1} \leftarrow R_k Q_k = Q_k^* T_k Q_k (Q_k^* \text{is the adjoint matrix of } Q_k)(k = 1, \cdots)$$

If $T_1, T_2, \cdots, T_k, T_{k+1}$ are created, they are all tridiagonal matrices. As $k \to \infty, T_k$ converges to a diagonal matrix having the eigenvalues of T as its diagonal elements.
To accelerate convergence in the actual QR method, the values μ_k are taken as approximations of the eigenvalues, $T_k - \mu_k I$ are created in place of T_k by performing an origin shift, and these are decomposed into:

$$T_k - \mu_k I = Q_k R_k$$

To calculate the approximation of an eigenvalue, consider the case when there is an adjacent eigenvalue (or an eigenvalue having a close absolute value). Let the eigenvalue μ_k of the lower-right corner submatrix obtained by the root-free QR method. If $T_{k+1} = R_k Q_k + \mu_k I$ is created, then T_{k+1} becomes:

$$T_{k+1} = Q_k^* T_k Q_k$$

After this operation is iterated until the sequence of matrices converges, the values adjusted by the cumulative amount the origin was shifted become the eigenvalues.

The eigenvectors of the original matrix before tridiagonalization are obtained by the following procedures. First, sequentially accumulate the transformation matrices used when obtaining the trigonal matrix T according to the Householder transformation method. Next, accumulate the transformation matrices $Q_1, Q_2, \cdots, Q_k$ obtained according to the QR method.

4.1.2.10 Root-free QR method

The root-free QR method, which eliminates the square root calculations of the QR method, is faster when only seeking the eigenvalues of a real symmetric tridiagonal matrix. Let $\alpha_1, \cdots, \alpha_n$ be the diagonal elements and $\beta_1, \cdots, \beta_{n-1}$ be the subdiagonal elements. Let one of the components of the transformation matrix during the calculation be $P^{(i)}$ and let S_i and C_i be $\sin\theta$ and $\cos\theta$ within $P^{(i)}$. Although square roots must be computed in the calculations:

$$P_i = \alpha_i C_{i-1} - \beta_{i-1} S_{i-1} C_{i-2}$$

$$S_i = \frac{\beta_i}{\sqrt{P_i^2 + \beta_i^2}}$$

$$C_i = \frac{P_i}{\sqrt{P_i^2 + \beta_i^2}}$$

$$\text{New } \alpha_{i-1} = \alpha_i + P_{i-1} C_{i-2} - P_i C_{i-1}$$

$$\text{New } \beta_{i-2} = S_{i-2} \sqrt{P_{i-1}^2 + \beta_{i-1}^2}$$

If these calculations are performed using the squared formats of each of $P_i, \beta_i, S_i, andC_i$, and if the following values are assumed: $C_0 = 1, S_0 = 0, r_1 = 2, P_1^2 = \alpha_1^2, \alpha_{m+1} = \beta_{m+1} = 0$ then the equations can be expressed as follows:

$$t_i^2 = P_i^2 + \beta_{i+1}^2$$

$$\text{New } \beta_i^2 = S_{i-1}^2 t_i^2$$

$$S_i^2 = \frac{\beta_{i+1}^2}{t_i^2}, C_i^2 = \frac{P_i^2}{t_i^2}$$

$$P_{i+1}^2 = \alpha_{i+1}^2 C_i^2 - 2\alpha_i + S_i^2 \gamma_i + \beta_{i+1}^2 S_i^2 C_{i-1}^2$$

$$\gamma_{i+1} = \alpha_{i+1} C_i^2 = S_i^2 \gamma_i$$

$$\text{New } \alpha_i = \alpha_{i+1} + \gamma_i - \gamma_{i+1}$$

and the calculations can be performed without computing any square roots.

For details, refer to (11) in the reference bibliography.

4.1.2.11 Bisection method

The bisection method obtains several of the largest or smallest eigenvalues of a real symmetric tridiagonal matrix T. If we let $d_1, d_2, \cdots, d_n$ be the diagonal components of T, let $s_1, s_2, \cdots, s_{n-1}$ be the subdiagonal components, let λ be a variable, and create the sequence of functions:

$$f_0(\lambda) = 1$$

$$f_1(\lambda) = d_1 - \lambda$$

$$f_i(\lambda) = (d_i - \lambda)f_{i-1}(\lambda) - s_{i-1}^2 f_{i-2}(\lambda) \quad (i = 2, \cdots, n)$$

then $f_0(\lambda), f_1(\lambda), \cdots, f_m(\lambda)$ is the Sturm sequence. That is, if we let $L(\lambda)$ be the number of non-matching signs for the successive sequence of functions for a given λ, then this $L(\lambda)$ is equal to the number of eigenvalues less than λ. To prevent overflow or underflow, if $g_i(\lambda)$ is actually defined as:

$$g_i(\lambda) = \frac{f_i(\lambda)}{f_{i-1}(\lambda)} \quad (i = 1, 2, \cdots, n)$$

$L(\lambda)$ becomes the number of $g_i(\lambda)$ that are negative. Furthermore, $g_i(\lambda)$ satisfies the following:

$$g_1(\lambda) = d_1 - \lambda$$

$$g_i = (d_i - \lambda) - \frac{s_{i-1}^2}{g_{i-1}(\lambda)} \quad (i = 2, \cdots, n)$$

If $g_{i-1}(\lambda)$=0, then $g_i(\lambda)$ is assumed to be:

$$g_i(\lambda) = (d_i - \lambda) - \frac{|s_{i-1}|}{\varepsilon} \quad (\varepsilon : \text{Units for determining error})$$

Assume that the eigenvalues of T satisfy:

$$\lambda_1 \le \lambda_2 \le \cdots\cdots \le \lambda_m$$

From the Gerschgorin theorem, the lower limit ($x_{\min}$) and upper limit ($x_{\max}$) of all the eigenvalues are given by:

$$x_{\max} = \text{MAX}(d_i + (|s_{i-1}| + |s_i|)) \quad (1 \le i \le n)$$

$$x_{\min} = \text{MIN}(d_i - (|s_{i-1}| + |s_i|)) \quad (1 \le i \le n)$$

where, $s_0 = s_n = 0$ is assumed.
We continue to make the eigenvalue existence range smaller by repeatedly subdividing the interval while counting the number of eigenvalues as described above, based on $x_{\min}$ and $x_{\max}$. In this way, both ends of the infinitesimal interval can be made to converge to a given eigenvalue.
For information about the Sturm sequence of functions and the Gerschgorin theorem, refer to entries (2) and (5) of the bibliography.

4.1.2.12 Accumulation of similarity (unitary) transformation by block algorithm

In seeking the eigenvectors of a real symmetric matrix using the QR method or the inverse iteration method, it is necessary to calculate the accumulation of the similarity (unitary) matrices that had already been computed in the preceding Householder transformation. It is very effective to apply the block algorithm to this accumulating procedure for getting better performance.

Let P_k be a transformation matrix that is obtained in Householder transformation which transform the real symmetric matrix to a tridiagonal matrix.

$$P_k = \boldsymbol{I} - \frac{\boldsymbol{u}_k \boldsymbol{u}_k^T}{H_k}$$

$$H_k = \frac{1}{2}\boldsymbol{u}_k^T \boldsymbol{u}_k$$

The accumulation of the transformation matrix P_k becomes to:

$$P_1 P_2 \cdots P_{n-2} = \boldsymbol{I} - \Sigma_{i=1}^{n-2} \boldsymbol{u}_i \boldsymbol{w}_i^T$$

where $\boldsymbol{w}_i^T$ is expressed by the following recurrence formula.

$$\boldsymbol{w}_{n-2}^T = \frac{\boldsymbol{u}_{n-2}^T}{H_{n-2}}$$

$$\boldsymbol{w}_i^T = \frac{\boldsymbol{u}_i^T - \sum_{j=i-1}^{n-2} (\boldsymbol{u}_i^T \boldsymbol{u}_j) \boldsymbol{w}_j^T}{H_i}$$

If we let $\boldsymbol{V}$ be the eigenvectors of a real symmetric tridiagonal matrix which are obtained with the QR method or the inverse iteration method. The eigenvectors $\boldsymbol{X}$ of the original matrix is obtained by the following.

$$\boldsymbol{X} = P_1 \cdots P_{n-2} \boldsymbol{V}$$

$$= \boldsymbol{V} - \sum_{i=1}^{n-2} \boldsymbol{u}_i \boldsymbol{w}_i^T \boldsymbol{V}$$

A product of a similarity (unitary) matrix P_k and the eigenvectors $\boldsymbol{V}$ is a rank-one update. Therefore, the accumulation of the transformation matrices can be obtained at a higher speed if the matrix updates are performed in the lump. If the original matrix is Hermitian, the transpose notation "^{T}" should be replaced with the Hermite conjugate notation "*".

4.1.2.13 Inverse iteration method

The inverse iteration method is used to obtain the eigenvector corresponding to the eigenvalues obtained by the root-free QR method or bisection method.

Assume the approximate value μ_k of a given eigenvalue λ_k of the real symmetric tridiagonal matrix T has been obtained. If a suitable initial vector $\boldsymbol{v_0}$ has been chosen at this time and the linear simultaneous equations:

$$(T - \mu_k I)\boldsymbol{v}_i = \boldsymbol{v}_{i-1} \quad (i = 1, 2, \cdots)$$

are iteratively solved, then if $\boldsymbol{v_i}$ satisfy the convergence conditions, they converge to the eigenvector.

To solve the simultaneous linear equations, partial pivoting is performed while using the Gauss method to perform an LU decomposition. Then forward elimination and back substitution are performed.

4.1.2.14 Generalized eigenvalue problem

A Cholesky decomposition of B is performed:

$$B = LL^*$$

in the generalized eigenvalue problem for a Hermitian matrix:

$$A\boldsymbol{x} = \lambda B\boldsymbol{x} \quad (A : \text{ Hermitian}, B : \text{ Positive Hermitian})$$

yielding:

$$(L^{-1}A(L^*)^{-1})(L^*\boldsymbol{x}) = \lambda(L^*\boldsymbol{x})$$

If we set:

$$P = L^{-1}A(L^*)^{-1}$$

$$L^*\boldsymbol{x} = \boldsymbol{y}$$

then the generalized eigenvalue problem is transformed to a standard eigenvalue problem for the Hermitian matrix P.

$$P\boldsymbol{y} = \lambda\boldsymbol{y}$$

The eigenvector of matrix A is given by:

$$\boldsymbol{x} = (L^*)^{-1}\boldsymbol{y}$$

Generalized eigenvalue problems for Hermitian matrix other than $A\boldsymbol{x} = \lambda B\boldsymbol{x}$ (B : Positive Hermitian) are classified into two cases by the position of positive Hermitian Matrix B as:

$$AB\boldsymbol{x} = \lambda\boldsymbol{x}$$

and

$$BA\boldsymbol{x} = \lambda\boldsymbol{x}$$

The eigenvalues λ and the eigenvectors x of these equations can be obtained by reducing them to standard eigenvalue problems using the following procedure:

① Apply the Cholesky decomposition to the positive matrix B as $B = L^*L$. (Where L is a lower triangle matrix.)

② $AB\boldsymbol{x} = \lambda\boldsymbol{x}$ is reduced to the eigenvalue problem for $C = LAL^*$, and the eigenvectors are obtained by multiplying the inverse of L.

③ $BA\boldsymbol{x} = \lambda\boldsymbol{x}$ is reduced to the eigenvalue problem for $C = LAL^*$, and the eigenvectors are obtained by multiplying L^*.

4.1.2.15 QZ method and the combination shift QZ method

For $A\boldsymbol{x} = \lambda B\boldsymbol{x}$ where A is a Hessenberg matrix and B is a regular upper triangular matrix, if we let $C = AB^{-1}$, then C also is a Hessenberg matrix.
Therefore, if we let:

$$C_1 = C$$

$$C_k = Q_k R_k \ (Q_k \ : \ \text{Unitary matrix}, R_k \ : \ \text{Upper triangular matrix})$$

$$C_{k+1} = R_k Q_k = Q_k^* C_k Q_k$$

and use the QR method to create $C_1, C_2, \cdots, C_k, C_{k+1}, \cdots$ these matrices are Hessenberg matrices that converge to an upper triangular matrix as $k \to \infty$.
In addition, we can select a unitary matrix Z so that:

$$\begin{array}{l} A_{k+1} = Q_k A_k Z_k \\ B_{k+1} = Q_k B_k Z_k \end{array} \left(\begin{array}{ll} A_{k+1} & : \ \text{Hessenberg matrix} \\ B_{k+1} & : \ \text{Upper triangular matrix} \end{array} \right)$$

Since:

$$A_{k+1}(B_{k+1})^{-1} = Q_k A_k Z_k Z_k^T B_k^{-1} Q_k^T = Q_k A_k B_k^{-1} Q_k^T = C_{k+1}$$

at this time, A_k converges to an upper triangular matrix, and if α_i are the diagonal elements of A and β_i are the diagonal elements of B then the eigenvalues are expressed by α_i/β_i. Combination shift QZ method
In a manner similar to the origin shift in the QR and double QR methods, you can also shift the origin in the QZ method in order to accelerate convergence. The combination shift QZ method uses a combination of the origin shift methods used in the QR and double QR methods. For details, refer to (9) and (10) in the reference bibliography.

4.1.2.16 Subspace method

The starting vector group:

$$X_0 = (\boldsymbol{x}_1^{(0)}, \boldsymbol{x}_2^{(0)}, \boldsymbol{x}_3^{(0)}, \boldsymbol{x}_4^{(0)} \cdots, \cdots, \boldsymbol{x}_m^{(0)}) \ (\boldsymbol{x}_i^{(0)}, \boldsymbol{x}_j^{(0)} \text{are independent})$$

is determined for $A\boldsymbol{x} = \lambda B\boldsymbol{x}$ (A: Real symmetric matrix, B: Positive symmetric matrix.) If we let:

$$Y_k = BX_k$$

$$AX_{k+1} = Y_k \quad (k \to \infty)$$

then the space E_k over which the $\boldsymbol{x}_i^{(k)}$ extend converge to the space E_∞ over which the eigenvectors $\boldsymbol{\phi}_i(i = 1, 2, \cdots, m)$ corresponding to the m eigenvalues $\lambda_i(i = 1, 2, \cdots, m)$ having smallest absolute values extend. (However, we assume that the $x_i^{(0)}$ are not orthogonal to E_∞.)
To speed up convergence, we let $AZ_k = Y_k$ and use a projection onto the space over which the $Z_i^{(k)}$ of matrices A and B extend.

$$A_k = Z_k^T A Z_k \quad (Z_k = (Z_1^{(k)}, Z_2^{(k)}, \cdots, Z_m^{(k)}))$$

$$B_k = Z_k^T B Z_k$$

If we obtain the eigenvalues and eigenvectors of A_k and B_k, improve Z_k, and let it be X_{k+1}, then X_{k+1} converges faster to the eigenvector $\boldsymbol{\phi} = (\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \cdots, \boldsymbol{\phi}_m)$ to be obtained.

$$A_k Q_k = B_k Q_k \Lambda_k \left(\begin{array}{l} \Lambda_k\text{: Diagonal matrix having eigenvalues corresponding to } A_k \text{ and } B_k \\ \quad\ \text{as diagonal components.} \\ Q_k\text{: Matrix having eigenvectors of } A_k \text{ and } B_k \text{ as column vectors.} \end{array} \right)$$

$$X_{k+1} = Z_k Q_k$$

$$X_{k+1} \to \Phi \ (k \to \infty) \quad \Phi = (\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \cdots, \boldsymbol{\phi}_m)$$

$$\Lambda_k \to \Lambda \ (k \to \infty) \quad \Lambda = \begin{bmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_m \end{bmatrix}$$

In addition, to obtain eigenvalues having maximum absolute values, let:

$$Y_k = AX_k$$

$$BZ_k = Y_k$$

$$X_{k+1} = Z_k Q_k$$

To prevent the norm from expanding or the vectors from becoming close to being parallel to each other at an intermediate iteration during the actual computation, the vectors are normalized and orthogonalized for each iteration.

In addition, to make the convergence speed proportional to $|\lambda_i/\lambda_m|$, more vectors are used in the iterative processing than the number of eigenvalues to be actually obtained.

4.1.2.17 Sturm sequence check

If n is the number of negative elements that appear as diagonal elements when $(A - \lambda B)$ is LDL^T-decomposed in the generalized eigenvalue problem $A\boldsymbol{x} = \lambda B\boldsymbol{x}$, then n corresponds to the number of eigenvalues smaller than λ_m. (However, assume that all eigenvalues are positive.)

4.1.2.18 Jacobi-Davidson method

To solve large sparse Hermitian eigenvalue problems numerically, variants of a method proposed by Davidson (18) are frequently applied. These solvers use a succession of subspaces where the update of the subspace exploits approximate inverses of the problem matrix, A. For A, $A = A^H$ or $A^* = A^T$ holds where A^* denotes A with complex conjugate elements and $A^H = (A^T)^*$ (transposed and complex conjugate). The basic idea is: Let $\mathbf{V}^k$ be a subspace of the whole n-dimensional space with an orthonormal basis $w_1^k, \cdots, w_m^k$ and W the matrix with columns w_j^k, $S := W^H A W$, $\bar{\lambda}_j^k$ the eigenvalues of S, and T a matrix with the eigenvectors of S as columns. The columns x_j^k of WT are approximations to eigenvectors of A with Ritz values $\bar{\lambda}_j^k = (x_j^k)^H A x_j^k$ that approximate eigenvalues of A. Let us assume that $\bar{\lambda}_{j_s}^k, \cdots, \bar{\lambda}_{j_{s+l-1}}^k \in [\lambda_{\text{lower}}, \lambda_{\text{upper}}]$. For $j \in j_s, \cdots, j_{s+l-1}$ define

$$q_j^k = (A - \bar{\lambda}_j^k I)\, x_j^k\,, \qquad r_j^k = (\bar{A} - \bar{\lambda}_j^k I)^{-1} q_j^k\,, \tag{4.1}$$

and $\mathbf{V}^{k+1} = \text{span}(\mathbf{V}^k \cup r_{j_s}^k \cup \cdots \cup r_{j_{s+l-1}}^k)$ where $\bar{A}$ is an easy to invert approximation to A ($\bar{A} = \text{diag}(A)$ in (18). Then $\mathbf{V}^{k+1}$ is an $(m+l)$-dimensional subspace of the whole n-dimensional space , and the repetition of the procedure above gives in general improved approximations to eigenvalues and -vectors. Restarting may increase efficiency. For good convergence, $\mathbf{V}^k$ has to contain crude approximations to all eigenvectors of A with eigenvalues smaller than λ_{lower} (cf. (18)). The approximate inverse must not be too accurate, otherwise the method stalls. The reason for this was investigated in (19) and leads to the Jacobi-Davidson (JD) method with an improved definition of r_j^k:

$$[(I - x_j^k\,(x_j^k)^H)\;(\bar{A} - \bar{\lambda}_j^k I)\;(I - x_j^k\,(x_j^k)^H)]\;r_j^k \;=\; q_j^k\,. \tag{4.2}$$

The projection $(I - x_j^k\,(x_j^k)^H)$ in (4.2) is not easy to incorporate into the matrix, but there is no need to do so, and solving (4.2) is only slightly more expensive than solving (4.1). The method converges quadratically for $\bar{A} = A$.

4.1.2.19 Preconditioning for Jacobi-Davidson method

The character of the JD method is determined by the approximation $\bar{A}$ to A. For obtaining an approximate solution of the preconditioning system (4.2), we may try an iterative approach (cf. (13), (14), (19), (20)). Here, a real symmetric or a complex Hermitian version of the QMR algorithm are used (cf. (15), (16), (17)) that are directly applied to the projected system (4.2) with $\bar{A} = A$. The control of the QMR iteration is as follows. Iteration is stopped when the current residual norm is smaller than the residual norm of QMR in the previous inner JD iteration. By controlling the QMR residual norms, we achieve that the preconditioning system (4.2) is solved in low accuracy in the beginning and in increasing accuracy in the course of the JD iteration. For a block version of JD, the residual norms of each preconditioning system (4.2) are separately controlled for each eigenvector to approximate since some eigenvector approximations are more difficult to obtain than others. This adapts the control to the properties of the matrix's spectrum.

Complex Hermitian QMR

```
$p^0 = q^0 = d^0 = s^0 = 0,\ \nu^1 = 1,\ \kappa^0 = -1,\ w^1 = v^1 = r^0 = b - Bx^0$
$\gamma^1 = \|v^1\|,\ \xi^1 = \gamma^1,\ \rho^1 = (w^1)^T v^1,\ \epsilon^1 = (B^* w^1)^T v^1,\ \mu^1 = 0, \tau^1 = \frac{\epsilon^1}{\rho^1}$
    $i = 1, 2, \cdots$
            $p^i = \frac{1}{\gamma^i} v^i - \mu^i p^{i-1}$
            $q^i = \frac{1}{\xi^i} B^* w^i - \frac{\gamma^i \mu^i}{\xi^i} q^{i-1}$
            $v^{i+1} = \boxed{Bp^i} - \frac{\tau^i}{\gamma^i} v^i$
            $w^{i+1} = q^i - \frac{\tau^i}{\xi^i} w^i$
      • if ( $\|r^{i-1}\| <$ tolerance ) then STOP
      • $\gamma^{i+1} = \|v^{i+1}\|$
      • $\xi^{i+1} = \|w^{i+1}\|$
      • $\rho^{i+1} = (w^{i+1})^T v^{i+1}$
      • $\epsilon^{i+1} = ( \boxed{B^* w^{i+1}} )^T v^{i+1}$
            $\mu^{i+1} = \frac{\gamma^i \xi^i \rho^{i+1}}{\gamma^{i+1} \tau^i \rho^i}$
            $\tau^{i+1} = \frac{\epsilon^{i+1}}{\rho^{i+1}} - \gamma^{i+1} \mu^{i+1}$
            $\theta^i = \frac{|\tau^i|^2 (1 - \nu^i)}{\nu^i |\tau^i|^2 + |\gamma^{i+1}|^2}$
            $\kappa^i = \frac{-\gamma^i (\tau^i)^* \kappa^{i-1}}{\nu^i |\tau^i|^2 + |\gamma^{i+1}|^2}$
            $\nu^{i+1} = \frac{\nu^i |\tau^i|^2}{\nu^i |\tau^i|^2 + |\gamma^{i+1}|^2}$
            $d^i = \theta^i d^{i-1} + \kappa^i p^i$
            $s^i = \theta^i s^{i-1} + \kappa^i B p^i$
            $x^i = x^{i-1} + d^i$
            $r^i = r^{i-1} - s^i$
```

The algorithm above shows the QMR iteration used to precondition JD for complex Hermitian matrices. The method is derived from the QMR variant described in (16). Within JD, the matrix B in the algorithm above corresponds to the matrix $[(I - x_j^k\,(x_j^k)^H)\ (A - \bar{\lambda}_j^k I)\ (I - x_j^k\,(x_j^k)^H)]$ of the preconditioning system (4.2). Per QMR iteration, two matrix-vector operations with B and B^* (marked by frames in the algorithm above) are performed since QMR bases on the non-Hermitian Lanczos algorithm that requires operations with B and $B^T = B^*$ but not with B^H (17). For real symmetric problems, only one matrix-vector operation per QMR iteration is necessary since then $q^i = Bp^i$ and thus $v^{i+1} = q^i - (\tau^i/\gamma^i)v^i$ hold. The only matrix-vector multiplication to compute per iteration is then Bw^{i+1}. Naturally, B is not computed element-wise from $[(I - x_j^k\,(x_j^k)^H)\,(A - \bar{\lambda}_j^k I)\,(I - x_j^k\,(x_j^k)^H)]$; the operation Bp^i, for instance, is split into vector-vector operations and one matrix-vector operation with A.

4.1.3 Reference Bibliography

(1) Wilkinson, J. H. and Reinsch, C. , "Handbook for Automatic Computation, Vol. II, Linear Algebra", Springer-Verlag, (1971).

(2) Wilkinson, J. H. , "The Algebraic Eigenvalue Problem", Clarendon Press, Oxford, (1965).

(3) Dongarra J. J. , Sorensen D. C. , and Hammarling A. J. , "Block reduction of matrices to condensed forms for eigenvalue computations", Journal of Computational and Applied Mathematics, Vol. 27, pp. 215-227(1989).

(4) Dongarra J. J. and van de Geijn R. A. , "Reduction to Condensed Form for the Eigenvalue Problem on Distributed Memory Architectures", LAPACK Working Note 30, pp. 1-12(1991).

(5) Francis, J. G. F. , "The QR transformation, I, II", Comput. J. 4, pp. 265-271, pp. 332-345(1961, 1962).

(6) Cuppen, J. J. M., "A Divide and Conquer Method for the Symmetric Tridiagonal Eigenproblem", Numer. Math. 36, pp. 177-195(1981).

(7) Gu, M. and Eisenstat, S. C., "A Stable and Efficient Algorithm for the rank-1 modification of the symmetric eigenproblem", SIAM J. Matrix Anal. Appl. 15, pp. 1266-1276(1994).

(8) Gu, M. and Eisenstat, S. C., "A Divide-and-Conquer Algorithm for the Symmetric Tridiagonal Eigenproblem", SIAM J. Matrix Anal. Appl. 16, pp. 172-191(1995).

(9) Moler, C. B and Stewart, G. W. , "An Algorithm for Generalized Matrix Eigenvalue Problems", SIAM Numerical Analysis, Vol. 10, No. 2, pp. 241-256(1973).

(10) Ward, R. C. , "The Combination Shift QZ Algorithm", SIAM Numerical Analysis, Vol. 12, No. 6, pp. 835-853(1973).

(11) Y. Beppu and I. Ninomiya, "HQRII—A Fast Diagonalization Subroutine", Computers and Chemistry Vol. 6(1982).

(12) Basermann, A. , "Parallel preconditioned solvers for large sparse Hermitian eigenvalue problems" In *VECPAR'98 - Third International Conference for Vector and Parallel Processing. Lecture Notes in Computer Science*, Dongarra J and Hernandez V (eds). Springer: Berlin, 1999; **1573**:72–85.

(13) Basermann, A. , Steffen, B. , "New Preconditioned Solvers for Large Sparse Eigenvalue Problems on Massively Parallel Computers" In: Proceedings of the Eighth SIAM Conference on Parallel Processing for Scientific Computing (CD-ROM). SIAM, Philadelphia (1997)

(14) Basermann, A., Steffen, B., "Preconditioned solvers for large eigenvalue problems on massively parallel computers and workstation clusters" In *Parallel Computing: Fundamentals, Applications and New Directions*, D'Hollander EH, Joubert GR, Peters FJ, Trottenberg U (eds). Elsevier Science B. V., 1998; 565–572.

(15) Basermann, A. , "QMR and TFQMR methods for sparse nonsymmetric problems on massively parallel systems" In *The Mathematics of Numerical Analysis. Series: Lectures in Applied Mathematics*, Renegar J, Shub M, Smale S (eds). AMS, 1996; **32**:59–76.

(16) Bücker, H. M. , Sauren, M. , "A Parallel Version of the Quasi-Minimal Residual Method Based on Coupled Two-Term Recurrences" In: Lecture Notes in Computer Science, Vol. 1184. Springer (1996) 157–165

(17) Freund, R. W. , Nachtigal, N. M. , “QMR: A Quasi-Minimal Residual Method for Non-Hermitian Linear Systems” Numer. Math. **60** (1991) 315–339

(18) Kosugi, N. , “Modifications of the Liu-Davidson Method for Obtaining One or Simultaneously Several Eigensolutions of a Large Real Symmetric Matrix” Comput. Phys. **55** (1984) 426–436

(19) Sleijpen GLG, van der Vorst HA. , “A Jacobi-Davidson iteration method for linear eigenvalue problems” *SIAM J. Matrix Anal. Appl.* 1996; **17**:401–425.

(20) Sleijpen GLG, van der Vorst HA, Meijerink E. , “Efficient expansion of subspaces in the Jacobi-Davidson method for standard and generalized eigenproblems” *ETNA* 1998; **7**:75–89.

(21) Lanczos, C. , “An Iteration Method for the Solution of the Eigenvalue Problem of Linear Differential and Integral Operators”, J. Res. Nat. Bur. Standards, B45 (1950) 255-282.

(22) Paige, C. C. , “Computational Variants of the Lanczos Method for the Eigenproblem”, J. Inst. Math. Appl., 10 (1972) 373-381.

(23) Simon, H. D. , “The Lanczos Algorithm with Partial Reorthogonalization”, Math. Comp. , 42 (1984) 115-142.

4.2 REAL MATRIX (TWO-DIMENSIONAL ARRAY TYPE) (REAL ARGUMENT TYPE)

4.2.1 DCGEAA, RCGEAA
All Eigenvalues and All Eigenvectors of a Real Matrix

(1) **Function**

DCGEAA or RCGEAA uses a basic similarity transformation and the double QR method to obtain all eigenvalues of the real matrix A (two-dimensional array type) and all corresponding eigenvectors.

(2) **Usage**

Double precision:

CALL DCGEAA (A, LNA, N, ER, EI, VE, LNV, IW1, W1, IERR)

Single precision:

CALL RCGEAA (A, LNA, N, ER, EI, VE, LNV, IW1, W1, IERR)

(3) **Arguments**

D:Double precision real Z:Double precision complex I: $\left\{\begin{array}{l}\text{INTEGER(4) as for 32bit Integer}\\ \text{INTEGER(8) as for 64bit Integer}\end{array}\right\}$

R:Single precision real C:Single precision complex

No.	Argument	Type	Size	Input/ Output	Contents
1	A	$\left\{\begin{array}{c}D\\R\end{array}\right\}$	LNA, N	Input	Real matrix A (two-dimensional array type).
				Output	Input-time contents are not retained.
2	LNA	I	1	Input	Adjustable dimension of array A.
3	N	I	1	Input	Order of matrix A.
4	ER	$\left\{\begin{array}{c}D\\R\end{array}\right\}$	N	Output	Real parts of eigenvalues (See Notes (a) and (b)).
5	EI	$\left\{\begin{array}{c}D\\R\end{array}\right\}$	N	Output	Imaginary parts of eigenvalues (See Notes (a) and (b)).
6	VE	$\left\{\begin{array}{c}D\\R\end{array}\right\}$	LNV, N	Output	Eigenvectors (See Notes (c) and (d)).
7	LNV	I	1	Input	Adjustable dimension of array VE.
8	IW1	I	N	Work	Work area
9	W1	$\left\{\begin{array}{c}D\\R\end{array}\right\}$	N	Work	Work area
10	IERR	I	1	Output	Error indicator

(4) **Restrictions**

(a) $0 < \mathrm{N} \leq \mathrm{LNA}, \mathrm{LNV}$

(5) **Error indicator**

IERR value	Meaning	Processing
0	Normal termination.	
1000	N was equal to 1.	$\mathrm{ER}(1) \leftarrow \mathrm{A}(1,1)$, $\mathrm{EI}(1) \leftarrow 0.0$ and $\mathrm{VE}(1,1) \leftarrow 1.0$ are performed.
3000	Restriction (a) was not satisfied.	Processing is aborted.
$5000 + i$	The sequence did not converge in the step where the eigenvalues obtained. $(1 \leq i \leq \mathrm{N})$	Eigenvalues correctly obtained by this time are entered in elements $(i + 1)$ through N of ER and EI. No eigenvector is obtained at this time.

(6) **Notes**

(a) Eigenvalue real parts are stored in ER and eigenvalue imaginary parts are stored in EI. If the j-th element eigenvalue at this time is a complex number, then its conjugate complex eigenvalue is stored in the $(j + 1)$-th element. However, the positive imaginary part is stored first.

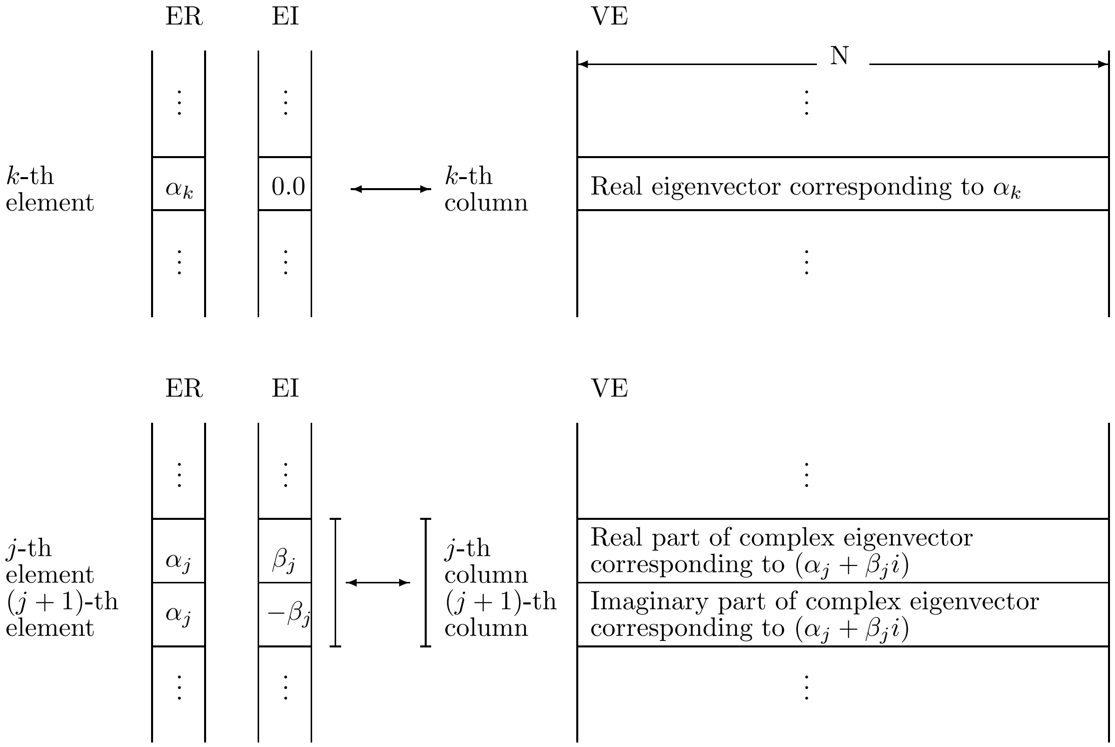
(b) Eigenvalues are obtained in decreasing order of their subscript values. That is, the j-th eigenvalue to be obtained is stored in the $(\mathrm{N} - j + 1)$-th element of ER and EI. However, the order in which eigenvalues are obtained is unrelated to the numerical values of the eigenvalues themselves.

(c) Eigenvectors are stored as shown in Figure 4−1 corresponding to eigenvalues (ER, EI). That is, if the k-th element eigenvalue is real, then the real eigenvector corresponding to it is stored in the k-th column of array VE. In addition, if the j-th and $(j + 1)$-th element eigenvalues are a pair of conjugate complex eigenvalues, then the real and imaginary parts of the complex eigenvector corresponding to the j-th element eigenvalue are stored in the j-th and $(j + 1)$-th columns respectively of array VE. The conjugate vector of this complex eigenvector becomes the eigenvector corresponding to the $(j + 1)$-th element eigenvalue.

(d) An eigenvector is normalized so that its Euclidean norm becomes $\|x\|_2 = 1.0$.

(e) If eigenvectors are not required, use 4.2.2 $\begin{Bmatrix} \mathrm{DCGEAN} \\ \mathrm{RCGEAN} \end{Bmatrix}$.

Figure 4−1 Eigenvalue and Eigenvector Storage Method


ER
EI
VE
N
k-th element
α_k
0.0
k-th column
Real eigenvector corresponding to α_k
ER
EI
VE
j-th element
$(j+1)$-th element
α_j
β_j
α_j
$-\beta_j$
j-th column
$(j+1)$-th column
Real part of complex eigenvector corresponding to $(\alpha_j+\beta_j i)$
Imaginary part of complex eigenvector corresponding to $(\alpha_j+\beta_j i)$

(7) **Example**

(a) Problem

Obtain all eigenvalues of the matrix:

$$A = \begin{bmatrix} 4 & -5 & 0 & 3 \\ 0 & 4 & -3 & -5 \\ 5 & -3 & 4 & 0 \\ 3 & 0 & 5 & 4 \end{bmatrix}$$

and their corresponding eigenvectors.

(b) Input data

Matrix A, LNA=11, N=4 and LNV=11.

(c) Main program

```
      PROGRAM BCGEAA
! *** EXAMPLE OF DCGEAA ***
      IMPLICIT REAL(8) (A-H,O-Z)
      PARAMETER ( ZERO = 0.0D0 )
      PARAMETER ( LNA = 11, LNV = 11 )
      DIMENSION A(LNA,LNA), ER(LNA), EI(LNA), VE(LNV,LNV),&
                IW1(LNA), W1(LNA)
!
      READ(5,*) N
      DO 10 I=1, N
         READ(5,*) (A(I,J), J=1, N)
   10 CONTINUE
!
      WRITE(6,1000) N
      DO 20 I=1, N
         WRITE(6,1100) (A(I,J), J=1, N)
   20 CONTINUE
!
      CALL DCGEAA(A,LNA,N,ER,EI,VE,LNV,IW1,W1,IERR)
!
      WRITE(6,1200)  IERR
!
      DO 70 J=1, N-1, 2
         WRITE(6,1300) ('EIGENVALUE ', I=1, 2)
         WRITE(6,1400) ER(J), EI(J), ER(J+1), EI(J+1)
         WRITE(6,1300) ('EIGENVECTOR', I=1, 2)
         IF(EI(J).EQ.ZERO) THEN
            IF(EI(J+1).EQ.ZERO) THEN
               DO 30 I=1, N
                  WRITE(6,1500) VE(I,J), ZERO, VE(I,J+1), ZERO
   30          CONTINUE
            ELSE
               DO 40 I=1, N
                  WRITE(6,1500) VE(I,J), ZERO, VE(I,J+1), VE(I,J+2)
   40          CONTINUE
            ENDIF
         ELSE
            IF(EI(J+1).EQ.ZERO) THEN
               DO 50 I=1, N
                  WRITE(6,1500) VE(I,J-1), -VE(I,J), VE(I,J+1), ZERO
   50          CONTINUE
            ELSE
               DO 60 I=1, N
                  WRITE(6,1500) VE(I,J), VE(I,J+1), VE(I,J), -VE(I,J+1)
   60          CONTINUE
            ENDIF
         ENDIF
   70 CONTINUE
      IF(MOD(N,2).NE.0) THEN
         WRITE(6,1300) 'EIGENVALUE '
         WRITE(6,1400) ER(N), EI(N)
         WRITE(6,1300) 'EIGENVECTOR'
         IF(EI(N).EQ.ZERO) THEN
            DO 80 I=1, N
               WRITE(6,1500) VE(I,N), ZERO
   80       CONTINUE
         ELSE
            DO 90 I=1, N
               WRITE(6,1500) VE(I,N-1), -VE(I,N)
   90       CONTINUE
         ENDIF
      ENDIF
      STOP
!
 1000 FORMAT(' ',/,/,&
             ' ***  DCGEAA  ***',/,/,&
             '  **  INPUT  **',/,/,&
             '      N = ', I2,/,/,&
             '      INPUT MATRIX A',/)
```

```
1100 FORMAT(7X, 11(F7.1))
1200 FORMAT(' ',/,/,&
            '  **  OUTPUT  **',/,/,&
            '      IERR = ', I4)
1300 FORMAT(' ',/, 2(14X, A11, 9X))
1400 FORMAT(' ', 2(4X, 1PD13.6, ' , ', 1PD13.6, 1X))
1500 FORMAT(' ', 2(4X, F13.10, ' , ', F13.10, 1X))
     END
```

(d) Output results

```
***  DCGEAA  ***

 **  INPUT  **

     N =  4

     INPUT MATRIX A

          4.0   -5.0    0.0    3.0
          0.0    4.0   -3.0   -5.0
          5.0   -3.0    4.0    0.0
          3.0    0.0    5.0    4.0


 **  OUTPUT  **

     IERR =    0

             EIGENVALUE                        EIGENVALUE
     1.200000D+01 ,  0.000000D+00      1.000000D+00 ,  5.000000D+00

             EIGENVECTOR                       EIGENVECTOR
     0.5000000000 ,  0.0000000000      0.1308649199 ,  0.4825705883
    -0.5000000000 ,  0.0000000000      0.4825705883 , -0.1308649199
     0.5000000000 ,  0.0000000000      0.4825705883 , -0.1308649199
     0.5000000000 ,  0.0000000000     -0.1308649199 , -0.4825705883

             EIGENVALUE                        EIGENVALUE
     1.000000D+00 , -5.000000D+00      2.000000D+00 ,  0.000000D+00

             EIGENVECTOR                       EIGENVECTOR
     0.1308649199 , -0.4825705883      0.5000000000 ,  0.0000000000
     0.4825705883 ,  0.1308649199      0.5000000000 ,  0.0000000000
     0.4825705883 ,  0.1308649199     -0.5000000000 ,  0.0000000000
    -0.1308649199 ,  0.4825705883      0.5000000000 ,  0.0000000000
```

4.2.2 DCGEAN, RCGEAN All Eigenvalues of a Real Matrix

(1) **Function**

DCGEAN or RCGEAN uses a basic similarity transformation and the double QR method to obtain all eigenvalues of the real matrix A (two-dimensional array type).

(2) **Usage**

Double precision:
CALL DCGEAN (A, LNA, N, ER, EI, IW1, W1, IERR)
Single precision:
CALL RCGEAN (A, LNA, N, ER, EI, IW1, W1, IERR)

(3) **Arguments**

D:Double precision real Z:Double precision complex I: $\left\{ \begin{array}{l} \text{INTEGER(4) as for 32bit Integer} \\ \text{INTEGER(8) as for 64bit Integer} \end{array} \right\}$
R:Single precision real C:Single precision complex

No.	Argument	Type	Size	Input/ Output	Contents
1	A	$\left\{\begin{matrix}\text{D}\\\text{R}\end{matrix}\right\}$	LNA, N	Input	Real matrix A (two-dimensional array type).
				Output	Input-time contents are not retained.
2	LNA	I	1	Input	Adjustable dimension of array A.
3	N	I	1	Input	Order of matrix A.
4	ER	$\left\{\begin{matrix}\text{D}\\\text{R}\end{matrix}\right\}$	N	Output	Real parts of eigenvalues (See Notes (a) and (b)).
5	EI	$\left\{\begin{matrix}\text{D}\\\text{R}\end{matrix}\right\}$	N	Output	Imaginary parts of eigenvalues (See Notes (a) and (b)).
6	IW1	I	N	Work	Work area
7	W1	$\left\{\begin{matrix}\text{D}\\\text{R}\end{matrix}\right\}$	N	Work	Work area
8	IERR	I	1	Output	Error indicator

(4) **Restrictions**

(a) $0 < \text{N} \leq \text{LNA}$

(5) **Error indicator**

IERR value	Meaning	Processing
0	Normal termination.	
1000	N was equal to 1.	ER(1) $\leftarrow$ A$(1,1)$ and EI(1) $\leftarrow$ 0.0 are performed.
3000	Restriction (a) was not satisfied.	Processing is aborted.
$5000+i$	The sequence did not converge in the step where the eigenvalues obtained. $(1 \leq i \leq \mathrm{N})$	Eigenvalues correctly obtained by this time are entered in elements $(i+1)$ through N of ER and EI.

(6) **Notes**

(a) Eigenvalue real parts are stored in ER and eigenvalue imaginary parts are stored in EI. If the j-th element eigenvalue at this time is a complex number, then its conjugate complex eigenvalue is stored in the $(j+1)$-th element. However, the positive imaginary part is stored first.

(b) Eigenvalues are obtained in decreasing order of their subscript values. That is, the j-th eigenvalue to be obtained is stored in the $(\mathrm{N}-j+1)$-th element of ER and EI. However, the order in which eigenvalues are obtained is unrelated to the numerical values of the eigenvalues themselves.

4.3 REAL MATRIX (TWO-DIMENSIONAL ARRAY TYPE) (COMPLEX ARGUMENT TYPE)

4.3.1 DCGNAA, RCGNAA All Eigenvalues and All Eigenvectors of a Real Matrix

(1) **Function**

DCGNAA or RCGNAA uses a basic similarity transformation and the double QR method to obtain all eigenvalues of the real matrix A (two-dimensional array type) and all corresponding eigenvectors.

(2) **Usage**

Double precision:
  CALL DCGNAA (A, LNA, N, E, VE, LNV, IW1, W1, IERR)
Single precision:
  CALL RCGNAA (A, LNA, N, E, VE, LNV, IW1, W1, IERR)

(3) **Arguments**

D:Double precision real Z:Double precision complex I: $\left\{\begin{matrix}\text{INTEGER(4) as for 32bit Integer}\\ \text{INTEGER(8) as for 64bit Integer}\end{matrix}\right\}$
R:Single precision real C:Single precision complex

No.	Argument	Type	Size	Input/ Output	Contents
1	A	$\left\{\begin{matrix}\text{D}\\ \text{R}\end{matrix}\right\}$	LNA, N	Input	Real matrix A (two-dimensional array type).
				Output	Input-time contents are not retained.
2	LNA	I	1	Input	Adjustable dimension of array A.
3	N	I	1	Input	Order of matrix A.
4	E	$\left\{\begin{matrix}\text{Z}\\ \text{C}\end{matrix}\right\}$	N	Output	Real parts of eigenvalues (See Notes (a) and (b)).
5	VE	$\left\{\begin{matrix}\text{Z}\\ \text{C}\end{matrix}\right\}$	LNV, N	Output	Eigenvectors (See Notes (c) and (d)).
6	LNV	I	1	Input	Adjustable dimension of array VE.
7	IW1	I	N	Work	Work area
8	W1	$\left\{\begin{matrix}\text{D}\\ \text{R}\end{matrix}\right\}$	N	Work	Work area
9	IERR	I	1	Output	Error indicator

(4) **Restrictions**

(a) $0 < \mathrm{N} \leq \mathrm{LNA}, \mathrm{LNV}$

(5) **Error indicator**

IERR value	Meaning	Processing
0	Normal termination.	
1000	N was equal to 1.	$\mathrm{E}(1) \leftarrow \mathrm{A}(1,1)$ and $\mathrm{VE}(1,1) \leftarrow 1.0$ are performed.
3000	Restriction (a) was not satisfied.	Processing is aborted.
$5000 + i$	The sequence did not converge in the step where the eigenvalues obtained. $(1 \leq i \leq \mathrm{N})$	Eigenvalues correctly obtained by this time are entered in elements $(i + 1)$ through N of E. No eigenvector is obtained at this time.

(6) **Notes**

(a) If the j-th element eigenvalue at this time is a complex number, then its conjugate complex eigenvalue is stored in the $(j + 1)$-th element. However, the positive imaginary part is stored first.

(b) Eigenvalues are obtained in decreasing order of their subscript values. That is, the j-th eigenvalue to be obtained is stored in the $(\mathrm{N} - j + 1)$-th element of E. However, the order in which eigenvalues are obtained is unrelated to the numerical values of the eigenvalues themselves.

(c) The eigenvector corresponding to the k-th element eigenvalue E is stored in the k-th column of array VE.

(d) An eigenvector is normalized so that its Euclidean norm becomes $\|x\|_2 = 1.0$.

(e) If eigenvectors are not required, use 4.3.2 $\left\{\begin{matrix}\mathrm{DCGNAN}\\ \mathrm{RCGNAN}\end{matrix}\right\}$.

(7) **Example**

(a) Problem
Obtain all eigenvalues of the matrix:

$$A = \begin{bmatrix} 4 & -5 & 0 & 3 \\ 0 & 4 & -3 & -5 \\ 5 & -3 & 4 & 0 \\ 3 & 0 & 5 & 4 \end{bmatrix}$$

and their corresponding eigenvectors.

(b) Input data
Matrix A, LNA=11, N=4 and LNV=11.

(c) Main program

```
      PROGRAM BCGNAA
! *** EXAMPLE OF DCGNAA ***
      IMPLICIT REAL(8) (A-H,O-Z)
      PARAMETER ( LNA = 11, LNV = 11 )
      COMPLEX(8) E,VE
      DIMENSION A(LNA,LNA),IW1(LNA),W1(LNA),E(LNA),VE(LNV,LNV)
!
      READ(5,*) N
      DO 10 I=1, N
         READ(5,*) (A(I,J), J=1, N)
   10 CONTINUE
!
      WRITE(6,1000) N
      DO 20 I=1, N
         WRITE(6,1100) (A(I,J), J=1, N)
   20 CONTINUE
!
      CALL DCGNAA(A,LNA,N,E,VE,LNV,IW1,W1,IERR)
!
      WRITE(6,1200)  IERR
!
      DO 70 J=1, N-1, 2
         WRITE(6,1300) ('EIGENVALUE ', I=1, 2)
         WRITE(6,1400) E(J),  E(J+1)
         WRITE(6,1300) ('EIGENVECTOR', I=1, 2)
               DO 30 I=1, N
                  WRITE(6,1500) VE(I,J),  VE(I,J+1)
   30          CONTINUE
   70 CONTINUE
      IF(MOD(N,2).NE.0) THEN
         WRITE(6,1300) 'EIGENVALUE '
         WRITE(6,1400) E(N)
         WRITE(6,1300) 'EIGENVECTOR'
            DO 80 I=1, N
               WRITE(6,1500) VE(I,N)
   80       CONTINUE
      ENDIF
      STOP
!
 1000 FORMAT(' ',/,/,&
             ' ***  DCGNAA  ***',/,/,&
             '  **  INPUT  **',/,/,&
             '      N = ', I2,/,/,&
             '      INPUT MATRIX A',/)
 1100 FORMAT(7X, 11(F7.1))
 1200 FORMAT(' ',/,/,&
             '  **  OUTPUT  **',/,/,&
             '      IERR = ', I4)
 1300 FORMAT(' ',/, 2(14X, A11, 9X))
 1400 FORMAT(' ', 2(4X, 1PD13.6, ' , ', 1PD13.6, 1X))
 1500 FORMAT(' ', 2(4X, F13.10, ' , ', F13.10, 1X))
      END
```

(d) Output results

```
***  DCGNAA  ***

 **  INPUT  **

     N =  4

     INPUT MATRIX A

          4.0   -5.0    0.0    3.0
          0.0    4.0   -3.0   -5.0
          5.0   -3.0    4.0    0.0
          3.0    0.0    5.0    4.0


 **  OUTPUT  **

     IERR =    0

              EIGENVALUE                        EIGENVALUE
     1.200000D+01 ,  0.000000D+00      1.000000D+00 ,  5.000000D+00

              EIGENVECTOR                       EIGENVECTOR
     0.5000000000 ,  0.0000000000      0.1308649199 ,  0.4825705883
    -0.5000000000 ,  0.0000000000      0.4825705883 , -0.1308649199
     0.5000000000 ,  0.0000000000      0.4825705883 , -0.1308649199
     0.5000000000 ,  0.0000000000     -0.1308649199 , -0.4825705883

              EIGENVALUE                        EIGENVALUE
     1.000000D+00 , -5.000000D+00      2.000000D+00 ,  0.000000D+00

              EIGENVECTOR                       EIGENVECTOR
     0.1308649199 , -0.4825705883      0.5000000000 ,  0.0000000000
     0.4825705883 ,  0.1308649199      0.5000000000 ,  0.0000000000
     0.4825705883 ,  0.1308649199     -0.5000000000 ,  0.0000000000
    -0.1308649199 ,  0.4825705883      0.5000000000 ,  0.0000000000
```

4.3.2 DCGNAN, RCGNAN All Eigenvalues of a Real Matrix

(1) **Function**

DCGNAN or RCGNAN uses a basic similarity transformation and the double QR method to obtain all eigenvalues of the real matrix A (two-dimensional array type).

(2) **Usage**

Double precision:

CALL DCGNAN (A, LNA, N, E, IW1, W1, IERR)

Single precision:

CALL RCGNAN (A, LNA, N, E, IW1, W1, IERR)

(3) **Arguments**

D:Double precision real Z:Double precision complex
R:Single precision real C:Single precision complex
I: $\left\{\begin{matrix}\text{INTEGER(4) as for 32bit Integer}\\\text{INTEGER(8) as for 64bit Integer}\end{matrix}\right\}$

No.	Argument	Type	Size	Input/ Output	Contents
1	A	$\left\{\begin{matrix}\mathrm{D}\\\mathrm{R}\end{matrix}\right\}$	LNA, N	Input	Real matrix A (two-dimensional array type).
				Output	Input-time contents are not retained.
2	LNA	I	1	Input	Adjustable dimension of array A.
3	N	I	1	Input	Order of matrix A.
4	E	$\left\{\begin{matrix}\mathrm{Z}\\\mathrm{C}\end{matrix}\right\}$	N	Output	Eigenvalues (See Notes (a) and (b)).
5	IW1	I	N	Work	Work area
6	W1	$\left\{\begin{matrix}\mathrm{D}\\\mathrm{R}\end{matrix}\right\}$	N	Work	Work area
7	IERR	I	1	Output	Error indicator

(4) **Restrictions**

(a) $0 < \mathrm{N} \leq \mathrm{LNA}$

(5) **Error indicator**

IERR value	Meaning	Processing
0	Normal termination.	
1000	N was equal to 1.	$\mathrm{E}(1) \leftarrow \mathrm{A}(1,1)$ is performed.
3000	Restriction (a) was not satisfied.	Processing is aborted.
$5000 + i$	The sequence did not converge in the step where the eigenvalues obtained. $(1 \leq i \leq \mathrm{N})$	Eigenvalues correctly obtained by this time are entered in elements $(i + 1)$ through N of E.

(6) **Notes**

(a) If the j-th element eigenvalue at this time is a complex number, then its conjugate complex eigenvalue is stored in the $(j + 1)$-th element. However, the positive imaginary part is stored first.

(b) Eigenvalues are obtained in decreasing order of their subscript values. That is, the j-th eigenvalue to be obtained is stored in the $(\mathrm{N} - j + 1)$-th element of E. However, the order in which eigenvalues are obtained is unrelated to the numerical values of the eigenvalues themselves.

4.4 COMPLEX MATRIX (TWO-DIMENSIONAL ARRAY TYPE) (REAL ARGUMENT TYPE)

4.4.1 ZCGEAA, CCGEAA
All Eigenvalues and All Eigenvectors of a Complex Matrix

(1) **Function**

ZCGEAA or CCGEAA uses a basic similarity transformation and QR method to obtain all eigenvalues of the complex matrix A=(AR, AI) (two-dimensional array type) and all corresponding eigenvectors.

(2) **Usage**

Double precision:
  CALL ZCGEAA (AR, AI, LNA, N, ER, EI, VR, VI, LNV, W1, IERR)
Single precision:
  CALL CCGEAA (AR, AI, LNA, N, ER, EI, VR, VI, LNV, W1, IERR)

(3) **Arguments**

D:Double precision real Z:Double precision complex I: $\left\{\begin{array}{l}\text{INTEGER(4) as for 32bit Integer}\\ \text{INTEGER(8) as for 64bit Integer}\end{array}\right\}$
R:Single precision real C:Single precision complex

No.	Argument	Type	Size	Input/ Output	Contents
1	AR	$\left\{\begin{array}{c}\text{D}\\ \text{R}\end{array}\right\}$	LNA, N	Input	Real part of complex matrix A (two-dimensional array type).
				Output	Input-time contents are not retained.
2	AI	$\left\{\begin{array}{c}\text{D}\\ \text{R}\end{array}\right\}$	LNA, N	Input	Imaginary part of complex matrix A (two-dimensional array type).
				Output	Input-time contents are not retained.
3	LNA	I	1	Input	Adjustable dimension of arrays AR and AI.
4	N	I	1	Input	Order of matrix A.
5	ER	$\left\{\begin{array}{c}\text{D}\\ \text{R}\end{array}\right\}$	N	Output	Real parts of eigenvalues (See Note (a)).
6	EI	$\left\{\begin{array}{c}\text{D}\\ \text{R}\end{array}\right\}$	N	Output	Imaginary parts of eigenvalues (See Note (a)).
7	VR	$\left\{\begin{array}{c}\text{D}\\ \text{R}\end{array}\right\}$	LNV, N	Output	Real parts (column vectors) of eigenvectors corresponding to eigenvalues (ER, EI) (See Notes (b) and (c)).
8	VI	$\left\{\begin{array}{c}\text{D}\\ \text{R}\end{array}\right\}$	LNV, N	Output	Imaginary parts (column vectors) of eigenvectors corresponding to eigenvalues (ER, EI) (See Notes (b) and (c)).
9	LNV	I	1	Input	Adjustable dimension of arrays VR and VI.

No.	Argument	Type	Size	Input/ Output	Contents
10	W1	$\begin{Bmatrix} \mathrm{D} \\ \mathrm{R} \end{Bmatrix}$	$3 \times \mathrm{N}$	Work	Work area
11	IERR	I	1	Output	Error indicator

(4) **Restrictions**

(a) $0 < \mathrm{N} \le \mathrm{LNA}, \mathrm{LNV}$

(5) **Error indicator**

IERR value	Meaning	Processing
0	Normal termination.	
1000	N was equal to 1.	$\mathrm{ER}(1) \leftarrow \mathrm{AR}(1,1)$, $\mathrm{EI}(1) \leftarrow \mathrm{AI}(1,1)$, $\mathrm{VR}(1,1) \leftarrow 1.0$ and $\mathrm{VI}(1,1) \leftarrow 0.0$ are performed.
3000	Restriction (a) was not satisfied.	Processing is aborted.
$5000 + i$	The sequence did not converge in the step where the eigenvalues obtained. $(1 \le i \le \mathrm{N})$	Eigenvalues correctly obtained by this time are entered in elements $(i + 1)$ through N of ER and EI. No eigenvector is obtained at this time.

(6) **Notes**

(a) Eigenvalue real parts are stored in ER and eigenvalue imaginary parts are stored in EI. Eigenvalues are obtained in decreasing order of their subscript values. That is, the j-th eigenvalue to be obtained is stored in the $(\mathrm{N} - j + 1)$-th element of ER and EI. However, the order in which eigenvalues are obtained is unrelated to the numerical values of the eigenvalues themselves.

(b) The real and imaginary parts of the eigenvector corresponding to the k-th element $(\mathrm{ER(k)}, \mathrm{EI(k)})$ of eigenvalue (ER, EI) are stored in the k-th columns of VR and VI respectively.

(c) An eigenvector is normalized so that its Euclidean norm becomes $\|x\|_2 = 1.0$.

(d) If eigenvectors are not required, use 4.4.2 $\begin{Bmatrix} \mathrm{ZCGEAN} \\ \mathrm{CCGEAN} \end{Bmatrix}$.

(7) Example

(a) problem

Obtain all eigenvalues of the matrix:

$$A = \begin{bmatrix} 5+9i & 5+5i & -6-6i & -7-7i \\ 3+3i & 6+10i & -5-5i & -6-6i \\ 2+2i & 3+3i & -1+3i & -5-5i \\ 1+i & 2+2i & -3-3i & 4i \end{bmatrix}$$

and their corresponding eigenvectors.

(b) Input data

Real part AR and imaginary AI of matrix A, LNA=11, N=4 and LNV=11.

(c) Main program

```
      PROGRAM ACGEAA
! *** EXAMPLE OF ZCGEAA ***
      IMPLICIT REAL(8)  (A-H,O-Z)
      CHARACTER*80 FMT
      PARAMETER ( LNA = 11, LNV = 11 )
      DIMENSION  AR(LNA,LNA),AI(LNA,LNA),ER(LNA),EI(LNA),&
                 VR(LNV,LNV),VI(LNV,LNV),W1(3*LNA)
!
      READ(5,*) N
      DO 10 I=1, N
         READ(5,*) (AR(I,J), AI(I,J), J=1, N)
   10 CONTINUE
!
      WRITE(6,1000) N
      DO 20 I=1, N
         WRITE(FMT,1100) N
         WRITE(6,FMT) (AR(I,J), AI(I,J), J=1, N)
   20 CONTINUE
!
      CALL ZCGEAA(AR,AI,LNA,N,ER,EI,VR,VI,LNV,W1,IERR)
!
      WRITE(6,1200) IERR
!
      DO 40 J=1, N-1, 2
         WRITE(6,1300) ('EIGENVALUE', I=1, 2)
         WRITE(6,1400) ER(J), EI(J), ER(J+1), EI(J+1)
         WRITE(6,1300) ('EIGENVECTOR', I=1, 2)
         DO 30 I=1, N
            WRITE(6,1500) VR(I,J), VI(I,J), VR(I,J+1), VI(I,J+1)
   30    CONTINUE
   40 CONTINUE
      IF(MOD(N,2).NE.0) THEN
         WRITE(6,1300) 'EIGENVALUE '
         WRITE(6,1400) ER(N), EI(N)
         WRITE(6,1300) 'EIGENVECTOR'
         DO 50 I=1, N
            WRITE(6,1500) VR(I,N), VI(I,N)
   50    CONTINUE
      ENDIF
      STOP
!
 1000 FORMAT(' ',/,/,&
             ' ***  ZCGEAA  ***',/,/,&
             '  **  INPUT  **',/,/,&
             '      N = ', I4,/,/,&
             '      INPUT MATRIX A ( REAL,IMAGINARY )',/)
 1100 FORMAT('( ,5X,',I2,'(''('',F5.1,'' , '', F5.1,'') ''))')
 1200 FORMAT(' ',/,/,&
             '  **  OUTPUT  **',/,/,&
             '      IERR = ', I4)
 1300 FORMAT(' ',/, 2(14X, A11, 9X))
 1400 FORMAT(' ', 2(4X, 1PD13.6, ' , ', 1PD13.6, 1X))
 1500 FORMAT(' ', 2(4X, F13.10, ' , ', F13.10, 1X))
      END
```

(d) Output results

```
***  ZCGEAA  ***

 **  INPUT  **

    N =    4

    INPUT MATRIX A ( REAL,IMAGINARY )

   (  5.0 ,   9.0) (  5.0 ,   5.0) ( -6.0 ,  -6.0) ( -7.0 ,  -7.0)
   (  3.0 ,   3.0) (  6.0 ,  10.0) ( -5.0 ,  -5.0) ( -6.0 ,  -6.0)
```

```
    (  2.0 ,   2.0) (  3.0 ,   3.0) ( -1.0 ,   3.0) ( -5.0 ,  -5.0)
    (  1.0 ,   1.0) (  2.0 ,   2.0) ( -3.0 ,  -3.0) (  0.0 ,   4.0)

**  OUTPUT  **

    IERR =    0

              EIGENVALUE                        EIGENVALUE
    4.000000D+00 ,  8.000000D+00      2.000000D+00 ,  6.000000D+00

             EIGENVECTOR                       EIGENVECTOR
    0.5419737851 , -0.1989918330      0.3438224256 , -0.1569817904
    0.5419737851 , -0.1989918330      0.6876448512 , -0.3139635808
    0.5419737851 , -0.1989918330      0.3438224256 , -0.1569817904
   -0.0000000000 ,  0.0000000000      0.3438224256 , -0.1569817904

              EIGENVALUE                        EIGENVALUE
    3.000000D+00 ,  7.000000D+00      1.000000D+00 ,  5.000000D+00

             EIGENVECTOR                       EIGENVECTOR
   -0.2921601411 ,  0.4979716712     -0.3883659462 ,  0.6485371718
   -0.2921601411 ,  0.4979716712     -0.1941829731 ,  0.3242685859
    0.0000000000 , -0.0000000000     -0.1941829731 ,  0.3242685859
   -0.2921601411 ,  0.4979716712     -0.1941829731 ,  0.3242685859
```

4.4.2 ZCGEAN, CCGEAN All Eigenvalues of a Complex Matrix

(1) **Function**

ZCGEAN or CCGEAN uses the a basic similarity transformation and QR method to obtain all eigenvalues of the complex matrix A=(AR, AI) (two-dimensional array type).

(2) **Usage**

Double precision:

CALL ZCGEAN (AR, AI, LNA, N, ER, EI, IERR)

Single precision:

CALL CCGEAN (AR, AI, LNA, N, ER, EI, IERR)

(3) **Arguments**

D:Double precision real Z:Double precision complex I: $\left\{ \begin{array}{l} \text{INTEGER(4) as for 32bit Integer} \\ \text{INTEGER(8) as for 64bit Integer} \end{array} \right\}$
R:Single precision real C:Single precision complex

No.	Argument	Type	Size	Input/ Output	Contents
1	AR	$\left\{\begin{array}{c}\text{D}\\\text{R}\end{array}\right\}$	LNA, N	Input	Real part of complex matrix A (two-dimensional array type).
				Output	Input-time contents are not retained.
2	AI	$\left\{\begin{array}{c}\text{D}\\\text{R}\end{array}\right\}$	LNA, N	Input	Imaginary part of complex matrix A (two-dimensional array type).
				Output	Input-time contents are not retained.
3	LNA	I	1	Input	Adjustable dimension of arrays AR and AI.
4	N	I	1	Input	Order of matrix A.
5	ER	$\left\{\begin{array}{c}\text{D}\\\text{R}\end{array}\right\}$	N	Output	Real parts of eigenvalues (See Note (a)).
6	EI	$\left\{\begin{array}{c}\text{D}\\\text{R}\end{array}\right\}$	N	Output	Imaginary parts of eigenvalues (See Note (a)).
7	IERR	I	1	Output	Error indicator

(4) **Restrictions**

(a) $0 < \text{N} \leq \text{LNA}$

(5) **Error indicator**

IERR value	Meaning	Processing
0	Normal termination.	
1000	N was equal to 1.	ER(1) ← AR$(1, 1)$ and EI(1) ← AI$(1, 1)$ are performed.
3000	Restriction (a) was not satisfied.	Processing is aborted.
$5000 + i$	The sequence did not converge in the step where the eigenvalues obtained. $(1 \leq i \leq \mathrm{N})$	Eigenvalues correctly obtained by this time are entered in elements $(i + 1)$ through N of ER and EI.

(6) **Notes**

(a) Eigenvalue real parts are stored in ER and eigenvalue imaginary parts are stored in EI. Eigenvalues are obtained in decreasing order of their subscript values. That is, the j-th eigenvalue to be obtained is stored in the $(\mathrm{N} - j + 1)$-th element of ER and EI. However, the order in which eigenvalues are obtained is unrelated to the numerical values of the eigenvalues themselves.

4.5 COMPLEX MATRIX (TWO-DIMENSIONAL ARRAY TYPE) (COMPLEX ARGUMENT TYPE)

4.5.1 ZCGNAA, CCGNAA All Eigenvalues and All Eigenvectors of a Complex Matrix

(1) **Function**

ZCGNAA or CCGNAA uses a basic similarity transformation and QR method to obtain all eigenvalues of the complex matrix A=(AR, AI) (two-dimensional array type) and all corresponding eigenvectors.

(2) **Usage**

Double precision:
CALL ZCGNAA (A, LNA, N, E, VE, LNV, W1, WK, IERR)
Single precision:
CALL CCGNAA (A, LNA, N, E, VE, LNV, W1, WK, IERR)

(3) **Arguments**

D:Double precision real Z:Double precision complex
R:Single precision real C:Single precision complex

I: $\left\{\begin{matrix}\text{INTEGER(4) as for 32bit Integer}\\ \text{INTEGER(8) as for 64bit Integer}\end{matrix}\right\}$

No.	Argument	Type	Size	Input/ Output	Contents
1	A	$\left\{\begin{matrix}\text{Z}\\ \text{C}\end{matrix}\right\}$	LNA, N	Input	Complex matrix A (two-dimensional array type).
				Output	Input-time contents are not retained.
2	LNA	I	1	Input	Adjustable dimension of array A.
3	N	I	1	Input	Order of matrix A.
4	E	$\left\{\begin{matrix}\text{Z}\\ \text{C}\end{matrix}\right\}$	N	Output	Eigenvalues (See Note (a)).
5	VE	$\left\{\begin{matrix}\text{Z}\\ \text{C}\end{matrix}\right\}$	LNV, N	Output	Eigenvectors (column vectors) corresponding to each eigenvalue (See Notes (b) and (c)).
6	LNV	I	1	Input	Adjustable dimension of array VE.
7	W1	$\left\{\begin{matrix}\text{D}\\ \text{R}\end{matrix}\right\}$	N	Work	Work area
8	WK	$\left\{\begin{matrix}\text{Z}\\ \text{C}\end{matrix}\right\}$	N	Work	Work area
9	IERR	I	1	Output	Error indicator

(4) **Restrictions**

(a) $0 < \text{N} \leq \text{LNA}, \text{LNV}$

(5) **Error indicator**

IERR value	Meaning	Processing
0	Normal termination.	
1000	N was equal to 1.	E(1) ← A(1, 1) and VE(1, 1) ← 1.0 are performed.
3000	Restriction (a) was not satisfied.	Processing is aborted.
$5000 + i$	The sequence did not converge in the step where the eigenvalues obtained. $(1 \le i \le \mathrm{N})$	Eigenvalues correctly obtained by this time are entered in elements $(i + 1)$ through N of E. No eigenvector is obtained at this time.

(6) **Notes**

(a) Eigenvalues are obtained in decreasing order of their subscript values. That is, the j-th eigenvalue to be obtained is stored in the $(\mathrm{N} - j + 1)$-th element of E. However, the order in which eigenvalues are obtained is unrelated to the numerical values of the eigenvalues themselves.

(b) The eigenvector corresponding to the k-th element eigenvalue E(k) are stored in the k-th columns of VE.

(c) An eigenvector is normalized so that its Euclidean norm becomes $\|x\|_2 = 1.0$.

(d) If eigenvectors are not required, use 4.5.2 $\left\{\begin{matrix}\text{ZCGNAN}\\ \text{CCGNAN}\end{matrix}\right\}$.

(7) **Example**

(a) problem

Obtain all eigenvalues of the matrix:

$$A = \begin{bmatrix} 5+9i & 5+5i & -6-6i & -7-7i \\ 3+3i & 6+10i & -5-5i & -6-6i \\ 2+2i & 3+3i & -1+3i & -5-5i \\ 1+i & 2+2i & -3-3i & 4i \end{bmatrix}$$

and their corresponding eigenvectors.

(b) Input data

Matrix A, LNA=11, N=4 and LNV=11.

(c) Main program

```
      PROGRAM ACGNAA
! *** EXAMPLE OF ZCGNAA ***
      IMPLICIT REAL(8)  (A-H,O-Z)
      CHARACTER*80 FMT
      PARAMETER ( LNA = 11, LNV = 11 )
      COMPLEX(8) A,E,VE,WK
      DIMENSION  A(LNA,LNA),E(LNA),VE(LNV,LNV),W1(LNA),WK(LNA)
!
      READ(5,*) N
      DO 10 I=1, N
         READ(5,*) (A(I,J), J=1, N)
   10 CONTINUE
!
      WRITE(6,1000) N
      DO 20 I=1, N
         WRITE(FMT,1100) N
         WRITE(6,FMT) (A(I,J), J=1, N)
   20 CONTINUE
!
```

```
      CALL ZCGNAA(A,LNA,N,E,VE,LNV,W1,WK,IERR)
!
      WRITE(6,1200) IERR
!
      DO 40 J=1, N-1, 2
         WRITE(6,1300) (’EIGENVALUE’, I=1, 2)
         WRITE(6,1400) E(J), E(J+1)
         WRITE(6,1300) (’EIGENVECTOR’, I=1, 2)
         DO 30 I=1, N
            WRITE(6,1500) VE(I,J), VE(I,J+1)
   30    CONTINUE
   40 CONTINUE
      IF(MOD(N,2).NE.0) THEN
         WRITE(6,1300) ’EIGENVALUE ’
         WRITE(6,1400) E(N)
         WRITE(6,1300) ’EIGENVECTOR’
         DO 50 I=1, N
            WRITE(6,1500) VE(I,N)
   50    CONTINUE
      ENDIF
      STOP
!
 1000 FORMAT(’ ’,/,/,&
             ’ ***  ZCGNAA  ***’,/,/,&
             ’  **  INPUT  **’,/,/,&
             ’      N = ’, I4,/,/,&
             ’      INPUT MATRIX A ( REAL,IMAGINARY )’,/)
 1100 FORMAT(’( ,5X,’,I2,’(’’(’’,F5.1,’’ , ’’, F5.1,’’) ’’))’)
 1200 FORMAT(’ ’,/,/,&
             ’  **  OUTPUT  **’,/,/,&
             ’      IERR = ’, I4)
 1300 FORMAT(’ ’,/, 2(14X, A11, 9X))
 1400 FORMAT(’ ’, 2(4X, 1PD13.6, ’ , ’, 1PD13.6, 1X))
 1500 FORMAT(’ ’, 2(4X, F13.10, ’ , ’, F13.10, 1X))
      END
```

(d) Output results

```
***  ZCGNAA  ***

 **  INPUT  **

     N =    4

     INPUT MATRIX A ( REAL,IMAGINARY )

    (  5.0 ,   9.0) (  5.0 ,   5.0) ( -6.0 ,  -6.0) ( -7.0 ,  -7.0)
    (  3.0 ,   3.0) (  6.0 ,  10.0) ( -5.0 ,  -5.0) ( -6.0 ,  -6.0)
    (  2.0 ,   2.0) (  3.0 ,   3.0) ( -1.0 ,   3.0) ( -5.0 ,  -5.0)
    (  1.0 ,   1.0) (  2.0 ,   2.0) ( -3.0 ,  -3.0) (  0.0 ,   4.0)


 **  OUTPUT  **

     IERR =    0

              EIGENVALUE                        EIGENVALUE
     4.000000D+00 ,  8.000000D+00      2.000000D+00 ,  6.000000D+00

             EIGENVECTOR                       EIGENVECTOR
     0.5419737851 , -0.1989918330     -0.3496783544 , -0.1434649481
     0.5419737851 , -0.1989918330     -0.6993567087 , -0.2869298963
     0.5419737851 , -0.1989918330     -0.3496783544 , -0.1434649481
     0.0000000000 ,  0.0000000000     -0.3496783544 , -0.1434649481

              EIGENVALUE                        EIGENVALUE
     3.000000D+00 ,  7.000000D+00      1.000000D+00 ,  5.000000D+00

             EIGENVECTOR                       EIGENVECTOR
    -0.2921601411 ,  0.4979716712     -0.3172488495 ,  0.6861353649
    -0.2921601411 ,  0.4979716712     -0.1586244247 ,  0.3430676824
     0.0000000000 ,  0.0000000000     -0.1586244247 ,  0.3430676824
    -0.2921601411 ,  0.4979716712     -0.1586244247 ,  0.3430676824
```

4.5.2 ZCGNAN, CCGNAN All Eigenvalues of a Complex Matrix

(1) **Function**

ZCGNAN or CCGNAN uses the a basic similarity transformation and QR method to obtain all eigenvalues of the complex matrix A=(AR, AI) (two-dimensional array type).

(2) **Usage**

Double precision:
CALL ZCGNAN (A, LNA, N, E, W1, IERR)
Single precision:
CALL CCGNAN (A, LNA, N, E, W1, IERR)

(3) **Arguments**

D:Double precision real Z:Double precision complex
R:Single precision real C:Single precision complex
I: $\left\{\begin{matrix}\text{INTEGER(4) as for 32bit Integer}\\ \text{INTEGER(8) as for 64bit Integer}\end{matrix}\right\}$

No.	Argument	Type	Size	Input/ Output	Contents
1	A	$\begin{Bmatrix}\mathrm{Z}\\ \mathrm{C}\end{Bmatrix}$	LNA, N	Input	Complex matrix A (two-dimensional array type).
				Output	Input-time contents are not retained.
2	LNA	I	1	Input	Adjustable dimension of array A.
3	N	I	1	Input	Order of matrix A.
4	E	$\begin{Bmatrix}\mathrm{Z}\\ \mathrm{C}\end{Bmatrix}$	N	Output	Eigenvalues (See Note (a)).
5	W1	$\begin{Bmatrix}\mathrm{D}\\ \mathrm{R}\end{Bmatrix}$	N	Work	Work area
6	IERR	I	1	Output	Error indicator

(4) **Restrictions**

(a) $0 < \mathrm{N} \leq \mathrm{LNA}$

(5) **Error indicator**

IERR value	Meaning	Processing
0	Normal termination.	
1000	N was equal to 1.	$\mathrm{E}(1) \leftarrow \mathrm{A}(1,1)$ is performed.
3000	Restriction (a) was not satisfied.	Processing is aborted.
$5000 + i$	The sequence did not converge in the step where the eigenvalues obtained. $(1 \leq i \leq \mathrm{N})$	Eigenvalues correctly obtained by this time are entered in elements $(i + 1)$ through N of E.

(6) **Notes**

(a) Eigenvalues are obtained in decreasing order of their subscript values. That is, the j-th eigenvalue to be obtained is stored in the $(\mathrm{N} - j + 1)$-th element of ER and EI. However, the order in which eigenvalues are obtained is unrelated to the numerical values of the eigenvalues themselves.

4.6 REAL SYMMETRIC MATRIX (TWO-DIMENSIONAL ARRAY TYPE) (UPPER TRIANGULAR TYPE)

4.6.1 DCSMAA, RCSMAA
All Eigenvalues and All Eigenvectors of a Real Symmetric Matrix

(1) **Function**

DCSMAA or RCSMAA uses the Householder method and QR method to obtain all eigenvalues of the real symmetric matrix A (two-dimensional array type) (upper triangular type) and all corresponding eigenvectors.

(2) **Usage**

Double precision:

CALL DCSMAA (A, LNA, N, E, W1, IERR)

Single precision:

CALL RCSMAA (A, LNA, N, E, W1, IERR)

(3) **Arguments**

D:Double precision real Z:Double precision complex I: $\left\{\begin{matrix}\text{INTEGER(4) as for 32bit Integer}\\\text{INTEGER(8) as for 64bit Integer}\end{matrix}\right\}$

R:Single precision real C:Single precision complex

No.	Argument	Type	Size	Input/ Output	Contents
1	A	$\begin{Bmatrix}\text{D}\\\text{R}\end{Bmatrix}$	LNA, N	Input	Real symmetric matrix A (two-dimensional array type) (upper triangular type)
				Output	Eigenvectors (column vectors) corresponding to each eigenvalue
2	LNA	I	1	Input	Adjustable dimension of array A
3	N	I	1	Input	Order of matrix A
4	E	$\begin{Bmatrix}\text{D}\\\text{R}\end{Bmatrix}$	N	Output	Eigenvalues
5	W1	$\begin{Bmatrix}\text{D}\\\text{R}\end{Bmatrix}$	N	Work	Work area
6	IERR	I	1	Output	Error indicator

(4) **Restrictions**

(a) $0 < \text{N} \leq \text{LNA}$

(5) **Error indicator**

IERR value	Meaning	Processing
0	Normal termination.	
1000	N was equal to 1.	E(1) ← A(1, 1) and A(1, 1) ← 1.0 are performed.
3000	Restriction (a) was not satisfied.	Processing is aborted.
$5000 + i$	The sequence did not converge in the step where the eigenvalues obtained. ($1 \leq i \leq$ N)	Eigenvalues correctly obtained by this time are entered in E(1), $\cdots$, E(i − 1) and eigenvectors corresponding to them are entered in A (However, the order is irregular).

(6) **Notes**

(a) Data should be stored only in the upper triangular portion of array A.

(b) Eigenvalues are stored in ascending order.

(c) The eigenvectors are an orthonormal system.

(d) If eigenvectors are not required, use 4.6.2 $\left\{\begin{matrix}\text{DCSMAN}\\\text{RCSMAN}\end{matrix}\right\}$.

(7) **Example**

(a) Problem
Obtain all eigenvalues of the matrix:

$$A = \begin{bmatrix} 6 & 4 & 4 & 1 \\ 4 & 6 & 1 & 4 \\ 4 & 1 & 6 & 4 \\ 1 & 4 & 4 & 6 \end{bmatrix}$$

and their corresponding eigenvectors.

(b) Input data
Matrix A, LNA=11 and N=4.

(c) Main program

```
      PROGRAM BCSMAA
! *** EXAMPLE OF DCSMAA ***
      IMPLICIT REAL(8) (A-H,O-Z)
      PARAMETER ( LNA = 11 )
      DIMENSION A(LNA,LNA), E(LNA), W1(LNA)
!
      READ(5,*) N
      DO 10 I=1, N
         READ(5,*) (A(I,J), J=I, N)
   10 CONTINUE
!
      WRITE(6,1000) N
      DO 20 I=1, N
         WRITE(6,1100) (A(J,I), J=1, I-1), (A(I,J), J=I, N)
   20 CONTINUE
!
      CALL DCSMAA(A,LNA,N,E,W1,IERR)
!
      WRITE(6,1200) IERR
!
      DO 40 K=1, N-3, 4
         WRITE(6,1300) ('EIGENVALUE ', I=1, 4)
         WRITE(6,1400) (E(I), I=K, K+3)
         WRITE(6,1300) ('EIGENVECTOR', I=1, 4)
         DO 30 J=1, N
            WRITE(6,1500) (A(J,I), I=K, K+3)
   30    CONTINUE
   40 CONTINUE
      IF(MOD(N,4).NE.0) THEN
         WRITE(6,1300) ('EIGENVALUE ', I=N/4*4+1, N)
         WRITE(6,1400) (E(I), I=N/4*4+1, N)
         WRITE(6,1300) ('EIGENVECTOR', I=N/4*4+1, N)
         DO 50 J=1, N
            WRITE(6,1500) (A(J,I), I=N/4*4+1, N)
   50    CONTINUE
      ENDIF
      STOP
!
 1000 FORMAT(' ',/,/,&
             ' ***  DCSMAA  ***',/,/,&
             '  **  INPUT  **',/,/,&
             '      N = ', I2,/,/,&
             '      INPUT MATRIX A',/)
 1100 FORMAT(7X, 11(F7.1))
 1200 FORMAT(' ',/,/,&
             '  **  OUTPUT  **',/,/,&
             '      IERR = ', I4)
 1300 FORMAT(' ',/,1X, 4(5X, A11, 2X))
 1400 FORMAT(3X, 4(2X, 1PD14.7, 2X))
 1500 FORMAT(2X, 4(F14.8, 4X))
      END
```

(d) Output results

```
***  DCSMAA  ***

 **  INPUT  **

     N =  4

     INPUT MATRIX A

          6.0    4.0    4.0    1.0
          4.0    6.0    1.0    4.0
          4.0    1.0    6.0    4.0
          1.0    4.0    4.0    6.0


 **  OUTPUT  **

     IERR =    0

     EIGENVALUE         EIGENVALUE         EIGENVALUE         EIGENVALUE
    -1.0000000D+00      5.0000000D+00      5.0000000D+00      1.5000000D+01

     EIGENVECTOR        EIGENVECTOR        EIGENVECTOR        EIGENVECTOR
     0.50000000         0.70710678         0.00000000         0.50000000
    -0.50000000         0.00000000        -0.70710678         0.50000000
    -0.50000000        -0.00000000         0.70710678         0.50000000
     0.50000000        -0.70710678         0.00000000         0.50000000
```

4.6.2 DCSMAN, RCSMAN
All Eigenvalues of a Real Symmetric Matrix

(1) **Function**

DCSMAN or RCSMAN uses the Householder method and root-free QR method to obtain all eigenvalues of the real symmetric matrix A (two-dimensional array type) (upper triangular type).

(2) **Usage**

Double precision:
  CALL DCSMAN (A, LNA, N, E, W1, IERR)
Single precision:
  CALL RCSMAN (A, LNA, N, E, W1, IERR)

(3) **Arguments**

D:Double precision real	Z:Double precision complex	I: $\left\{\begin{array}{l}\text{INTEGER(4) as for 32bit Integer}\\ \text{INTEGER(8) as for 64bit Integer}\end{array}\right\}$
R:Single precision real	C:Single precision complex	

No.	Argument	Type	Size	Input/ Output	Contents
1	A	$\left\{\begin{array}{c}D\\R\end{array}\right\}$	LNA, N	Input	Real symmetric matrix A (two-dimensional array type) (upper triangular type)
				Output	Input-time contents are not retained.
2	LNA	I	1	Input	Adjustable dimension of array A
3	N	I	1	Input	Order of matrix A
4	E	$\left\{\begin{array}{c}D\\R\end{array}\right\}$	N	Output	Eigenvalues
5	W1	$\left\{\begin{array}{c}D\\R\end{array}\right\}$	N	Work	Work area
6	IERR	I	1	Output	Error indicator

(4) **Restrictions**

(a) $0 < \mathrm{N} \leq \mathrm{LNA}$

(5) **Error indicator**

IERR value	Meaning	Processing
0	Normal termination.	
1000	N was equal to 1.	$\mathrm{E}(1) \leftarrow \mathrm{A}(1,1)$ is performed.
3000	Restriction (a) was not satisfied.	Processing is aborted.
$5000 + i$	The sequence did not converge in the step where the eigenvalues obtained. ($1 \leq i \leq \mathrm{N}$)	Eigenvalues correctly obtained by this time are entered in $\mathrm{E}(1), \cdots, \mathrm{E}(\mathrm{i}-1)$ (However, the order is irregular).

(6) **Notes**

(a) Data should be stored only in the upper triangular portion of array A.

(b) Eigenvalues are stored in ascending order.

4.6.3 DCSMSS, RCSMSS Eigenvalues and Eigenvectors of a Real Symmetric Matrix

(1) **Function**

DCSMSS or RCSMSS uses the Householder method, root free QR method, or Bisection method to obtain the m largest or m smallest eigenvalues of the real symmetric matrix A (two-dimensional array type) (upper triangular type) and the inverse iteration method to obtain the corresponding eigenvectors.

(2) **Usage**

Double precision:
CALL DCSMSS (A, LNA, N, EPS, E, M, VE, LNV, ISW, IW1, W1, IERR)
Single precision:
CALL RCSMSS (A, LNA, N, EPS, E, M, VE, LNV, ISW, IW1, W1, IERR)

(3) **Arguments**

D:Double precision real Z:Double precision complex
R:Single precision real C:Single precision complex
I: $\left\{\begin{array}{l}\text{INTEGER(4) as for 32bit Integer}\\ \text{INTEGER(8) as for 64bit Integer}\end{array}\right\}$

No.	Argument	Type	Size	Input/ Output	Contents
1	A	$\left\{\begin{array}{c}D\\R\end{array}\right\}$	LNA, N	Input	Real symmetric matrix A (two-dimensional array type) (upper triangular type)
				Output	Input-time contents are not retained.
2	LNA	I	1	Input	Adjustable dimension of array A
3	N	I	1	Input	Order of matrix A
4	EPS	$\left\{\begin{array}{c}D\\R\end{array}\right\}$	1	Input	Parameter that assigns an upper limit to the absolute error for use in the eigenvalues convergence test. (See Note (d))
5	E	$\left\{\begin{array}{c}D\\R\end{array}\right\}$	M	Output	Eigenvalues
6	M	I	1	Input	The number m of eigenvalues to be obtained.
7	VE	$\left\{\begin{array}{c}D\\R\end{array}\right\}$	LNV, M	Output	Eigenvectors (column vector) corresponding to each eigenvalue.
8	LNV	I	1	Input	Adjustable dimension of array VE
9	ISW	I	1	Input	Processing switch ISW≥0: Obtain M eigenvalues from the largest one. ISW<0: Obtain M eigenvalues from the smallest one.

No.	Argument	Type	Size	Input/ Output	Contents
10	IW1	I	M	Output	Eigenvectors flag (See Note (e))
11	W1	$\begin{Bmatrix}\mathrm{D}\\\mathrm{R}\end{Bmatrix}$	$8 \times \mathrm{N}$	Work	Work area
12	IERR	I	1	Output	Error indicator

(4) **Restrictions**

(a) $0 < \mathrm{N} \leq \mathrm{LNA}, \mathrm{LNV}$

(b) $0 < \mathrm{M} \leq \mathrm{N}$

(5) **Error indicator**

IERR value	Meaning	Processing
0	Normal termination.	
1000	N was equal to 1.	$\mathrm{E}(1) \leftarrow \mathrm{A}(1,1)$ and $\mathrm{VE}(1,1) \leftarrow 1.0$ are performed.
2000	The maximum number of iterations was exceeded by the inverse iterations for obtaining eigenvectors.	Some eigenvectors are obtained with low precision, and processing continues. (See Note (e).)
3000	Restriction (a) or (b) was not satisfied.	Processing is aborted.

(6) **Notes**

(a) Data should be stored only in the upper triangular portion of array A.

(b) If ISW≥0, the eigenvalues are stored in descending order. If ISW<0, they are stored in ascending order.

(c) Eigenvalue calculations are appropriately divided up between the root-free QR method and Bisection method internally.

(d) If EPS ≤ 0, the optimum value is automatically set internally. Normally, a negative value should be set so that this value will be set automatically. EPS is used to obtain eigenvalues by using the Bisection method.

(e) If the maximum number of iterations is exceeded when using the inverse iteration method (IERR = 2000 is output), the following processing is performed.
If IW1(i) = 0: The i-th eigenvector calculation is normally terminated.
If IW1(i) $\neq$ 0: The convergence condition is not satisfied for the i-th eigenvector calculation, and the eigenvector precision is low. In this case, the iteration count is set for IW1(i).
If processing is normally terminated (IERR = 0 is output), IW1(i) = 0 is set.

(f) The eigenvectors are an orthonormal system.

(g) If eigenvectors are not required, use 4.6.4 $\begin{Bmatrix}\mathrm{DCSMSN}\\\mathrm{RCSMSN}\end{Bmatrix}$.

(7) **Example**

(a) Problem

Obtain the three smallest eigenvalues of the matrix:

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

and their corresponding eigenvectors.

(b) Input data

Matrix A, LNA=11, N=6, EPS=−1.0, M=3, LNV=11 and ISW=−1.

(c) Main program

```
      PROGRAM BCSMSS
! *** EXAMPLE OF DCSMSS ***
      IMPLICIT REAL(8) (A-H,O-Z)
      PARAMETER ( LNA = 11, LNV = 11 )
      DIMENSION A(LNA,LNA), E(LNA), VE(LNV,LNV), IW1(LNA), W1(8*LNA)
!
      READ(5,*) N, M
      DO 10 I=1, N
         READ(5,*) (A(I,J), J=I, N)
   10 CONTINUE
!
      WRITE(6,1000) N, M
      DO 20 I=1, N
         WRITE(6,1100) (A(J,I), J=1, I-1), (A(I,J), J=I, N)
   20 CONTINUE
!
      ISW = -1
      EPS = -1.0D0
!
      CALL DCSMSS(A,LNA,N,EPS,E,M,VE,LNV,ISW,IW1,W1,IERR)
!
      WRITE(6,1200) IERR
!
      DO 40 K=1, M-3, 4
         WRITE(6,1300) ('EIGENVALUE ', I=1, 4)
         WRITE(6,1400) (E(I), I=K, K+3)
         WRITE(6,1300) ('EIGENVECTOR', I=1, 4)
         DO 30 J=1, N
            WRITE(6,1500) (VE(J,I), I=K, K+3)
   30    CONTINUE
   40 CONTINUE
      IF(MOD(M,4).NE.0) THEN
         WRITE(6,1300) ('EIGENVALUE ', I=M/4*4+1, M)
         WRITE(6,1400) (E(I), I=M/4*4+1, M)
         WRITE(6,1300) ('EIGENVECTOR', I=M/4*4+1, M)
         DO 50 J=1, N
            WRITE(6,1500) (VE(J,I), I=M/4*4+1, M)
   50    CONTINUE
      ENDIF
      STOP
!
 1000 FORMAT(' ',/,/,&
             ' ***  DCSMSS  ***',/,/,&
             '  **  INPUT  **',/,/,&
             '      N = ', I2,/,/,&
             '      M = ', I2,/,/,&
             '      INPUT MATRIX A',/)
 1100 FORMAT(7X, 11(F7.1))
 1200 FORMAT(' ',/,/,&
             '  **  OUTPUT  **',/,/,&
             '      IERR = ', I4)
 1300 FORMAT(' ',/,1X, 4(5X, A11, 2X))
 1400 FORMAT(3X, 4(2X, 1PD14.7, 2X))
 1500 FORMAT(2X, 4(F14.8, 4X))
      END
```

(d) Output results

```
*** DCSMSS ***

 ** INPUT **

    N =  6

    M =  3

    INPUT MATRIX A

         0.0    1.0    0.0    0.0    0.0    1.0
         1.0    0.0    1.0    0.0    0.0    0.0
         0.0    1.0    0.0    1.0    0.0    0.0
         0.0    0.0    1.0    0.0    1.0    0.0
         0.0    0.0    0.0    1.0    0.0    1.0
         1.0    0.0    0.0    0.0    1.0    0.0

 ** OUTPUT **

    IERR =    0

    EIGENVALUE        EIGENVALUE        EIGENVALUE
   -2.0000000D+00    -1.0000000D+00    -1.0000000D+00

    EIGENVECTOR       EIGENVECTOR       EIGENVECTOR
    0.40824829        0.00000000        0.57735027
   -0.40824829       -0.50000000       -0.28867513
    0.40824829        0.50000000       -0.28867513
   -0.40824829       -0.00000000        0.57735027
    0.40824829       -0.50000000       -0.28867513
   -0.40824829        0.50000000       -0.28867513
```

4.6.4 DCSMSN, RCSMSN Eigenvalues of a Real Symmetric Matrix

(1) **Function**

DCSMSN or RCSMSN uses the Householder method, root-free QR method, or Bisection method to obtain the m largest or m smallest eigenvalues of the real symmetric matrix A (two-dimensional array type) (upper triangular type).

(2) **Usage**

Double precision:
CALL DCSMSN (A, LNA, N, EPS, E, M, ISW, W1, IERR)
Single precision:
CALL RCSMSN (A, LNA, N, EPS, E, M, ISW, W1, IERR)

(3) **Arguments**

D:Double precision real Z:Double precision complex I: $\left\{\begin{matrix}\text{INTEGER(4) as for 32bit Integer}\\\text{INTEGER(8) as for 64bit Integer}\end{matrix}\right\}$
R:Single precision real C:Single precision complex

No.	Argument	Type	Size	Input/ Output	Contents
1	A	$\begin{Bmatrix}\text{D}\\\text{R}\end{Bmatrix}$	LNA, N	Input	Real symmetric matrix A (two-dimensional array type) (upper triangular type)
				Output	Input-time contents are not retained.
2	LNA	I	1	Input	Adjustable dimension of array A
3	N	I	1	Input	Order of matrix A
4	EPS	$\begin{Bmatrix}\text{D}\\\text{R}\end{Bmatrix}$	1	Input	Parameter that assigns an upper limit to the absolute error for use in the eigenvalues convergence test. (See Note (d))
5	E	$\begin{Bmatrix}\text{D}\\\text{R}\end{Bmatrix}$	M	Output	Eigenvalues
6	M	I	1	Input	The number m of eigenvalues to be obtained.
7	ISW	I	1	Input	Processing switch ISW≥0: Obtain M eigenvalues from the largest one. ISW<0: Obtain M eigenvalues from the smallest one.
8	W1	$\begin{Bmatrix}\text{D}\\\text{R}\end{Bmatrix}$	5 × N	Work	Work area
9	IERR	I	1	Output	Error indicator

(4) **Restrictions**

(a) $0 < \text{N} \leq \text{LNA}$

(b) $0 < \text{M} \leq \text{N}$

(5) **Error indicator**

IERR value	Meaning	Processing
0	Normal termination.	
1000	N was equal to 1.	$\mathrm{E}(1) \leftarrow \mathrm{A}(1,1)$ is performed.
3000	Restriction (a) or (b) was not satisfied.	Processing is aborted.

(6) **Notes**

(a) Data should be stored only in the upper triangular portion of array A.

(b) If ISW$\geq$0, the eigenvalues are stored in descending order. If ISW<0, they are stored in ascending order.

(c) Eigenvalue calculations are appropriately divided up between the root-free QR method and Bisection method internally.

(d) If EPS ≤ 0, the optimum value is automatically set internally. Normally, a negative value should be set so that this value will be set automatically. EPS is used to obtain eigenvalues by using the Bisection method.

4.6.5 DCSMEE, RCSMEE
Eigenvalues in an Interval and Their Eigenvectors of a Real Symmetric Matrix (Interval Specified)

(1) **Function**

DCSMEE or RCSMEE uses the Householder method and the Bisection method to obtain the m largest or m smallest eigenvalues in a specified interval of the real symmetric matrix A (two-dimensional array type)(upper triangular type) and the inverse iteration method to obtain the corresponding eigenvectors.

(2) **Usage**

Double precision:

CALL DCSMEE (A, LNA, N, EPS, E, M, E1, E2, VE, LNV, IW1, W1, IERR)

Single precision:

CALL RCSMEE (A, LNA, N, EPS, E, M, E1, E2, VE, LNV, IW1, W1, IERR)

(3) **Arguments**

D:Double precision real Z:Double precision complex
R:Single precision real C:Single precision complex
I: $\begin{Bmatrix}\text{INTEGER(4) as for 32bit Integer}\\\text{INTEGER(8) as for 64bit Integer}\end{Bmatrix}$

No.	Argument	Type	Size	Input/ Output	Contents
1	A	$\begin{Bmatrix}\text{D}\\\text{R}\end{Bmatrix}$	LNA, N	Input	Real symmetric matrix A (two-dimensional array type) (upper triangular type)
				Output	Input-time contents are not retained.
2	LNA	I	1	Input	Adjustable dimension of array A
3	N	I	1	Input	Order of matrix A
4	EPS	$\begin{Bmatrix}\text{D}\\\text{R}\end{Bmatrix}$	1	Input	Parameter that assigns an upper limit to the absolute error for use in the eigenvalues convergence test. (See Note (b))
5	E	$\begin{Bmatrix}\text{D}\\\text{R}\end{Bmatrix}$	M	Output	Eigenvalues
6	M	I	1	Input	Maximum number of the eigenvalues to be computed
				Output	Number of the obtained eigenvalues
7	E1	$\begin{Bmatrix}\text{D}\\\text{R}\end{Bmatrix}$	1	Input	E1 < E2: Obtain M eigenvalues in the interval [E1, E2] from the smallest one. (E2 is upper bound.) E1>E2: Obtain M eigenvalues in the interval [E1, E2] from the largest one. (E2 is lower bound.) (See Notes (c) and (d))
8	E2	$\begin{Bmatrix}\text{D}\\\text{R}\end{Bmatrix}$	1	Input	

No.	Argument	Type	Size	Input/ Output	Contents
9	VE	$\left\{\begin{matrix}\text{D}\\\text{R}\end{matrix}\right\}$	LNV, M	Output	Eigenvectors (column vector) corresponding to each eigenvalue.
10	LNV	I	1	Input	Adjustable dimension of array VE
11	IW1	I	M	Output	Eigenvectors flag (See Note (e))
12	W1	$\left\{\begin{matrix}\text{D}\\\text{R}\end{matrix}\right\}$	8 × N	Work	Work area
13	IERR	I	1	Output	Error indicator

(4) **Restrictions**

(a) $0 < \text{N} \le \text{LNA}, \text{LNV}$

(b) $0 < \text{M} \le \text{N}$

(5) **Error indicator**

IERR value	Meaning	Processing
0	Normal termination.	
1000	N was equal to 1.	$\text{E}(1) \leftarrow \text{A}(1,1)$ and $\text{VE}(1,1) \leftarrow 1.0$ are performed.
1500	The number of eigenvalues between E1 and E2 is less than M.	All the eigenvalues and the corresponding eigenvectors between E1 and E2 are obtained and the number of the found eigenvalue is output to M.
2000	The maximum number of iterations was exceeded by the inverse iterations for obtaining eigenvectors.	Some eigenvectors are obtained with low precision, and processing continues. (See Note (e).)
3000	Restriction (a) or (b) was not satisfied.	Processing is aborted.

(6) **Notes**

(a) Data should be stored only in the upper triangular portion of array A.

(b) If EPS ≤ 0, the optimum value is automatically set internally. Normally, a negative value should be set so that this value will be set automatically. EPS is used to obtain eigenvalues by using the Bisection method.

(c) If E1 < E2 the obtained eigenvalues and eigenvectors are stored in ascending order. On the other hand, if E1>E2 the eigenvalues and eigenvectors are stored in descending order.

(d) If E1 = E2, the eigenvalues in the interval [E1 − EPS, E1 + EPS] are obtained. Normally, E1 should be set to be different E2.

(e) If the maximum number of iterations is exceeded when using the inverse iteration method (IERR = 2000 is output), the following processing is performed.
If IW1(i) = 0: The i-th eigenvector calculation is normally terminated.
If IW1(i) ≠ 0: The convergence condition is not satisfied for the i-th eigenvector calculation, and the

eigenvector precision is low. In this case, the iteration count is set for IW1(i). If processing is normally terminated (IERR = 0 is output), IW1(i) = 0 is set.

(f) The eigenvectors are an orthonormal system.

(g) If eigenvectors are not required, use 4.6.6 $\left\{\begin{matrix}\text{DCSMEN}\\\text{RCSMEN}\end{matrix}\right\}$.

(7) **Example**

(a) Problem
Obtain the three eigenvalues in the interval [0, 5] from the smallest one of the following symmetric matrix:

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

and their corresponding eigenvectors.

(b) Input data
Matrix A, LNA=11, N=6, EPS=-1.0, M=3, E1=0, E2=5 and LNV=10.

(c) Main program

```
      PROGRAM BCSMEE
! *** EXAMPLE OF DCSMEE ***
      IMPLICIT REAL(8) (A-H,O-Z)
      PARAMETER ( LNA = 11, LNV = 11 )
      DIMENSION A(LNA,LNA), E(LNA), VE(LNV,LNV), IW1(LNA), W1(8*LNA)
!
      READ(5,*) N, M, E1, E2
      DO 10 I=1, N
         READ(5,*) (A(I,J), J=I, N)
   10 CONTINUE
!
      WRITE(6,1000) N, M, E1, E2
      DO 20 I=1, N
         WRITE(6,1100) (A(J,I), J=1, I-1), (A(I,J), J=I, N)
   20 CONTINUE
!
      EPS = -1.0D0
!
      CALL DCSMEE(A,LNA,N,EPS,E,M,E1,E2,VE,LNV,IW1,W1,IERR)
!
      WRITE(6,1200) IERR
!
      DO 40 K=1, M-3, 4
         WRITE(6,1300) ('EIGENVALUE ', I=1, 4)
         WRITE(6,1400) (E(I), I=K, K+3)
         WRITE(6,1300) ('EIGENVECTOR', I=1, 4)
         DO 30 J=1, N
            WRITE(6,1500) (VE(J,I), I=K, K+3)
   30    CONTINUE
   40 CONTINUE
      IF(MOD(M,4).NE.0) THEN
         WRITE(6,1300) ('EIGENVALUE ', I=M/4*4+1, M)
         WRITE(6,1400) (E(I), I=M/4*4+1, M)
         WRITE(6,1300) ('EIGENVECTOR', I=M/4*4+1, M)
         DO 50 J=1, N
            WRITE(6,1500) (VE(J,I), I=M/4*4+1, M)
   50    CONTINUE
      ENDIF
      STOP
!
 1000 FORMAT(1X,/,/,&
             1X,'***  DCSMEE  ***',/,/,&
             1X,' **  INPUT  **',/,/,&
             1X,'N = ', I4,'  M = ', I4,/,/,&
             1X,'E1= ', 1PD14.7,'  E2= ', 1PD14.7,/,/,&
             1X,'     INPUT MATRIX A',/)
 1100 FORMAT(1X, 6X, 11(F7.1))
 1200 FORMAT(1X,/,/,&
             1X,' **  OUTPUT  **',/,/,&
             1X,'     IERR = ', I4)
 1300 FORMAT(1X,/,1X, 4(5X, A11, 2X))
 1400 FORMAT(1X, 2X, 4(2X, 1PD14.7, 2X))
 1500 FORMAT(1X, 1X, 4(F14.8, 4X))
      END
```

(d) Output results

```
***  DCSMEE  ***

 **  INPUT  **

N =    6  M =    3

E1=  0.0000000D+00  E2=  5.0000000D+00

     INPUT MATRIX A

          0.0    1.0    0.0    0.0    0.0    1.0
          1.0    0.0    1.0    0.0    0.0    0.0
          0.0    1.0    0.0    1.0    0.0    0.0
          0.0    0.0    1.0    0.0    1.0    0.0
          0.0    0.0    0.0    1.0    0.0    1.0
          1.0    0.0    0.0    0.0    1.0    0.0

 **  OUTPUT  **

     IERR =    0

     EIGENVALUE        EIGENVALUE        EIGENVALUE
     1.0000000D+00     1.0000000D+00      2.0000000D+00

     EIGENVECTOR       EIGENVECTOR       EIGENVECTOR
     0.57735027        0.00000000       -0.40824829
     0.28867513       -0.50000000       -0.40824829
    -0.28867513       -0.50000000       -0.40824829
    -0.57735027       -0.00000000       -0.40824829
    -0.28867513        0.50000000       -0.40824829
     0.28867513        0.50000000       -0.40824829
```

4.6.6 DCSMEN, RCSMEN
Eigenvalues in an Interval of a Real Symmetric Matrix (Interval Specified)

(1) **Function**

DCSMEN or RCSMEN uses the Householder method and the Bisection method to obtain m largest or m smallest eigenvalues in a specified interval of the real symmetric matrix A (two-dimensional array type) (upper triangular type).

(2) **Usage**

Double precision:

CALL DCSMEN (A, LNA, N, EPS, E, M, E1, E2 , W1, IERR)

Single precision:

CALL RCSMEN (A, LNA, N, EPS, E, M, E1, E2 , W1, IERR)

(3) **Arguments**

D:Double precision real	Z:Double precision complex	I: $\left\{\begin{array}{l}\text{INTEGER(4) as for 32bit Integer}\\\text{INTEGER(8) as for 64bit Integer}\end{array}\right\}$
R:Single precision real	C:Single precision complex	

No.	Argument	Type	Size	Input/ Output	Contents
1	A	$\left\{\begin{array}{c}\text{D}\\\text{R}\end{array}\right\}$	LNA, N	Input	Real symmetric matrix A (two-dimensional array type) (upper triangular type)
				Output	Input-time contents are not retained.
2	LNA	I	1	Input	Adjustable dimension of array A
3	N	I	1	Input	Order of matrix A
4	EPS	$\left\{\begin{array}{c}\text{D}\\\text{R}\end{array}\right\}$	1	Input	Parameter that assigns an upper limit to the absolute error for use in the eigenvalues convergence test. (See Note (b))
5	E	$\left\{\begin{array}{c}\text{D}\\\text{R}\end{array}\right\}$	M	Output	Eigenvalues
6	M	I	1	Input	Maximum number of the eigenvalues to be computed
				Output	Number of the obtained eigenvalues
7	E1	$\left\{\begin{array}{c}\text{D}\\\text{R}\end{array}\right\}$	1	Input	E1<E2: Obtain M eigenvalues in the interval [E1, E2] from the smallest one. (E2 is upper bound.) E1>E2: Obtain M eigenvalues in the interval [E1, E2] from the largest one. (E2 is lower bound.) (See Notes (c) and (d))
8	E2	$\left\{\begin{array}{c}\text{D}\\\text{R}\end{array}\right\}$	1	Input	
9	W1	$\left\{\begin{array}{c}\text{D}\\\text{R}\end{array}\right\}$	5 × N	Work	Work area
10	IERR	I	1	Output	Error indicator

(4) **Restrictions**

(a) $0 < \mathrm{N} \leq \mathrm{LNA}$

(b) $0 < \mathrm{M} \leq \mathrm{N}$

(5) **Error indicator**

IERR value	Meaning	Processing
0	Normal termination.	
1000	N was equal to 1.	$\mathrm{E}(1) \leftarrow \mathrm{A}(1,1)$ is performed.
1500	The number of eigenvalues between E1 and E2 is less than M.	All the eigenvalues and the corresponding eigenvectors between E1 and E2 are obtained and the number of the found eigenvalue is output to M.
3000	Restriction (a) or (b) was not satisfied.	Processing is aborted.

(6) **Notes**

(a) Data should be stored only in the upper triangular portion of array A.

(b) If $\mathrm{EPS} \leq 0$, the optimum value is automatically set internally. Normally, a negative value should be set so that this value will be set automatically. EPS is used to obtain eigenvalues by using the Bisection method.

(c) If $\mathrm{E1} < \mathrm{E2}$ the obtained eigenvalues and eigenvectors are stored in ascending order. On the other hand, if $\mathrm{E1} > \mathrm{E2}$ the eigenvalues and eigenvectors are stored in descending order.

(d) If $\mathrm{E1} = \mathrm{E2}$, the eigenvalues in the interval $[\mathrm{E1} - \mathrm{EPS}, \mathrm{E1} + \mathrm{EPS}]$ are obtained. Normally, E1 should be set to be different E2.

4.7 HERMITIAN MATRIX (TWO-DIMENSIONAL ARRAY TYPE) (UPPER TRIANGULAR TYPE) (REAL ARGUMENT TYPE)

4.7.1 ZCHRAA, CCHRAA
All Eigenvalues and All Eigenvectors of a Hermitian Matrix

(1) **Function**

ZCHRAA or CCHRAA uses the Householder method and QR method to obtain all eigenvalues of the Hermitian matrix A=(AR, AI) (two-dimensional array type) (upper triangular type) (real argument type) and all corresponding eigenvectors.

(2) **Usage**

Double precision:
  CALL ZCHRAA (AR, AI, LNA, N, E, VR, VI, LNV, W1, IERR)
Single precision:
  CALL CCHRAA (AR, AI, LNA, N, E, VR, VI, LNV, W1, IERR)

(3) **Arguments**

D:Double precision real	Z:Double precision complex	I: $\left\{\begin{matrix}\text{INTEGER(4) as for 32bit Integer}\\ \text{INTEGER(8) as for 64bit Integer}\end{matrix}\right\}$
R:Single precision real	C:Single precision complex	

No.	Argument	Type	Size	Input/ Output	Contents
1	AR	$\left\{\begin{matrix}\text{D}\\ \text{R}\end{matrix}\right\}$	LNA, N	Input	Real part of Hermitian matrix A (two-dimensional array type) (upper triangular type)
2	AI	$\left\{\begin{matrix}\text{D}\\ \text{R}\end{matrix}\right\}$	LNA, N	Input	Imaginary part of Hermitian matrix A (two-dimensional array type) (upper triangular type)
3	LNA	I	1	Input	Adjustable dimension of arrays AR and AI
4	N	I	1	Input	Order of matrix A
5	E	$\left\{\begin{matrix}\text{D}\\ \text{R}\end{matrix}\right\}$	N	Output	Eigenvalues
6	VR	$\left\{\begin{matrix}\text{D}\\ \text{R}\end{matrix}\right\}$	LNV, N	Output	Real part (column vector) of eigenvectors corresponding to each eigenvalue
7	VI	$\left\{\begin{matrix}\text{D}\\ \text{R}\end{matrix}\right\}$	LNV, N	Output	Imaginary part (column vectors) of eigenvectors corresponding to each eigenvalue
8	LNV	I	1	Input	Adjustable dimension of arrays VR and VI
9	W1	$\left\{\begin{matrix}\text{D}\\ \text{R}\end{matrix}\right\}$	$3 \times \text{N}$	Work	Work area
10	IERR	I	1	Output	Error indicator

(4) **Restrictions**

(a) $0 < \text{N} \leq \text{LNA}, \text{LNV}$

(5) **Error indicator**

IERR value	Meaning	Processing
0	Normal termination.	
1000	N was equal to 1.	$\mathrm{E}(1) \leftarrow \mathrm{A}(1,1)$, $\mathrm{VR}(1,1) \leftarrow 1.0$ and $\mathrm{VI}(1,1) \leftarrow 0.0$ are performed.
3000	Restriction (a) was not satisfied.	Processing is aborted.
$5000+i$	The sequence did not converge in the step where the eigenvalue is obtained. $(1 \leq i \leq \mathrm{N})$	Eigenvalues obtained by this time are entered in $\mathrm{E}(1), \cdots, \mathrm{E}(\mathrm{i}-1)$ (However, the order is irregular). Not eigenvector is obtained at this time.

(6) **Notes**

(a) Real and imaginary parts of the Hermitian matrix are stored only in the upper triangular portions of arrays AR and AI respectively. (See Appendix B)

(b) Eigenvalues are stored in ascending order.

(c) The eigenvectors are an orthonormal set.

(d) If eigenvectors are not required, use 4.7.2 $\left\{\begin{matrix}\mathrm{ZCHRAN}\\ \mathrm{CCHRAN}\end{matrix}\right\}$.

(7) **Example**

(a) Problem

Obtain all eigenvalues of the matrix:

$$A = \begin{bmatrix} 7 & 3 & 1+2i & -1+2i \\ 3 & 7 & 1-2i & -1-2i \\ 1-2i & 1+2i & 7 & -3 \\ -1-2i & -1+2i & -3 & 7 \end{bmatrix}$$

and their corresponding eigenvectors.

(b) Input data

Real part AR and imaginary part AI of matrix A, LNA=11, N=4 and LNV=10.

(c) Main program

```
      PROGRAM ACHRAA
! *** EXAMPLE OF ZCHRAA ***
      IMPLICIT REAL(8) (A-H,O-Z)
      CHARACTER*80 FMT
      PARAMETER ( LNA = 11, LNV = 11 )
      DIMENSION AR(LNA,LNA), AI(LNA,LNA), E(LNA),&
                VR(LNV,LNV), VI(LNV,LNV), W1(3*LNA)
!
      READ(5,*) N
      DO 10 I=1, N
         READ(5,*) (AR(I,J), AI(I,J), J=I, N)
   10 CONTINUE
!
      WRITE(6,1000) N
      DO 20 I=1, N
         WRITE(FMT,1100) N
         WRITE(6,FMT) (AR(J,I), -AI(J,I), J=1, I-1),&
                      (AR(I,J),  AI(I,J), J=I, N)
   20 CONTINUE
!
      CALL ZCHRAA(AR,AI,LNA,N,E,VR,VI,LNV,W1,IERR)
!
```

```
      WRITE(6,1200) IERR
!
      DO 40 J=1, N-1, 2
         WRITE(6,1300) ('EIGENVALUE ', I=1, 2)
         WRITE(6,1400) E(J), E(J+1)
         WRITE(6,1300) ('EIGENVECTOR', I=1, 2)
         DO 30 I=1, N
            WRITE(6,1500) VR(I,J), VI(I,J), VR(I,J+1), VI(I,J+1)
   30    CONTINUE
   40 CONTINUE
      IF(MOD(N,2).NE.0) THEN
         WRITE(6,1300) 'EIGENVALUE '
         WRITE(6,1400) E(N)
         WRITE(6,1300) 'EIGENVECTOR'
         DO 50 I=1, N
            WRITE(6,1500) VR(I,N), VI(I,N)
   50    CONTINUE
      ENDIF
      STOP
!
 1000 FORMAT(' ',/,/,&
             ' ***  ZCHRAA  ***',/,/,&
             '  **  INPUT  **',/,/,&
             '      N = ', I4,/,/,&
             '      INPUT MATRIX A ( REAL,IMAGINARY )',/)
 1100 FORMAT('( ,5X,', I2,'(''('',F5.1,'' ,'',F5.1,'') ''))')
 1200 FORMAT(' ',/,/,&
             '  **  OUTPUT  **',/,/,&
             '      IERR = ', I4)
 1300 FORMAT(' ',/, 2(14X, A11, 8X))
 1400 FORMAT(' ', 2(12X, 1PD14.7, 7X))
 1500 FORMAT(' ', 2(5X, F12.8, ' ,', F12.8, 2X))
      END
```

(d) Output results

```
***  ZCHRAA  ***

 **  INPUT  **

     N =    4

     INPUT MATRIX A ( REAL,IMAGINARY )

    (  7.0 ,   0.0) (  3.0 ,   0.0) (  1.0 ,   2.0) ( -1.0 ,   2.0)
    (  3.0 ,   0.0) (  7.0 ,   0.0) (  1.0 ,  -2.0) ( -1.0 ,  -2.0)
    (  1.0 ,  -2.0) (  1.0 ,   2.0) (  7.0 ,   0.0) ( -3.0 ,   0.0)
    ( -1.0 ,  -2.0) ( -1.0 ,   2.0) ( -3.0 ,   0.0) (  7.0 ,   0.0)


 **  OUTPUT  **

     IERR =    0

             EIGENVALUE                       EIGENVALUE
             0.0000000D+00                    8.0000000D+00

             EIGENVECTOR                      EIGENVECTOR
       0.50000000 ,  0.00000000        -0.70710678 , -0.00000000
      -0.50000000 ,  0.00000000         0.00000000 , -0.00000000
       0.00000000 ,  0.50000000         0.35355339 ,  0.35355339
      -0.00000000 ,  0.50000000        -0.35355339 ,  0.35355339

             EIGENVALUE                       EIGENVALUE
             8.0000000D+00                    1.2000000D+01

             EIGENVECTOR                      EIGENVECTOR
       0.00000000 ,  0.00000000         0.50000000 ,  0.00000000
      -0.09987868 ,  0.70001732         0.50000000 ,  0.00000000
      -0.30006932 , -0.39994800         0.50000000 , -0.00000000
      -0.39994800 ,  0.30006932        -0.50000000 , -0.00000000
```

4.7.2 ZCHRAN, CCHRAN
All Eigenvalues of a Hermitian Matrix

(1) **Function**

ZCHRAN or CCHRAN uses the Householder method and root-free QR method to obtain all eigenvalues of the Hermitian matrix A=(AR, AI) (two-dimensional array type) (upper triangular type) (real argument type).

(2) **Usage**

Double precision:

CALL ZCHRAN (AR, AI, LNA, N, E, W1, IERR)

Single precision:

CALL CCHRAN (AR, AI, LNA, N, E, W1, IERR)

(3) **Arguments**

D:Double precision real Z:Double precision complex I: $\left\{\begin{matrix}\text{INTEGER(4) as for 32bit Integer}\\\text{INTEGER(8) as for 64bit Integer}\end{matrix}\right\}$
R:Single precision real C:Single precision complex

No.	Argument	Type	Size	Input/ Output	Contents
1	AR	$\begin{Bmatrix}\text{D}\\\text{R}\end{Bmatrix}$	LNA, N	Input	Real part of Hermitian matrix A (two-dimensional array type) (upper triangular type)
				Output	Input-time contents are not retained.
2	AI	$\begin{Bmatrix}\text{D}\\\text{R}\end{Bmatrix}$	LNA, N	Input	Imaginary part of Hermitian matrix A (two-dimensional array type) (upper triangular type)
				Output	Input-time contents are not retained.
3	LNA	I	1	Input	Adjustable dimension of arrays AR and AI
4	N	I	1	Input	Order of matrix A
5	E	$\begin{Bmatrix}\text{D}\\\text{R}\end{Bmatrix}$	N	Output	Eigenvalues
6	W1	$\begin{Bmatrix}\text{D}\\\text{R}\end{Bmatrix}$	$3 \times$ N	Work	Work area
7	IERR	I	1	Output	Error indicator

(4) **Restrictions**

(a) $0 <$ N $\leq$ LNA

(5) **Error indicator**

IERR value	Meaning	Processing
0	Normal termination.	
1000	N was equal to 1.	E(1) ← AR(1, 1) is performed.
3000	Restriction (a) was not satisfied.	Processing is aborted.
$5000 + i$	The sequence did not converge in the step where the eigenvalue is obtained. ($1 \leq i \leq$ N)	Eigenvalues obtained by this time are entered in E(1), $\cdots$, E(i − 1) (However, the order is irregular).

(6) **Notes**

(a) Real and imaginary parts of the Hermitian matrix are stored only in the upper triangular portions of arrays AR and AI respectively. (See Appendix B)

(b) Eigenvalues are stored in ascending order.

4.7.3 ZCHRSS, CCHRSS Eigenvalues and Eigenvectors of a Hermitian Matrix

(1) **Function**

ZCHRSS or CCHRSS uses the Householder method, root-free QR method, or Bisection method to obtain the m largest or m smallest eigenvalues of the Hermitian matrix A=(AR, AI) (two-dimensional array type) (upper triangular type) (real argument type) and the inverse iteration method to obtain the corresponding eigenvectors.

(2) **Usage**

Double precision:

CALL ZCHRSS (AR, AI, LNA, N, EPS, E, M, VR, VI, LNV, ISW, IW1, W1, IERR)

Single precision:

CALL CCHRSS (AR, AI, LNA, N, EPS, E, M, VR, VI, LNV, ISW, IW1, W1, IERR)

(3) **Arguments**

D:Double precision real Z:Double precision complex I: $\left\{\begin{array}{l}\text{INTEGER(4) as for 32bit Integer}\\\text{INTEGER(8) as for 64bit Integer}\end{array}\right\}$
R:Single precision real C:Single precision complex

No.	Argument	Type	Size	Input/ Output	Contents
1	AR	$\left\{\begin{array}{c}\text{D}\\\text{R}\end{array}\right\}$	LNA, N	Input	Real part of Hermitian matrix A (two-dimensional array type) (upper triangular type)
				Output	Input-time contents are not retained.
2	AI	$\left\{\begin{array}{c}\text{D}\\\text{R}\end{array}\right\}$	LNA, N	Input	Imaginary part of Hermitian matrix A (two-dimensional array type) (upper triangular type)
				Output	Input-time contents are not retained.
3	LNA	I	1	Input	Adjustable dimension of arrays AR and AI
4	N	I	1	Input	Order of matrix A
5	EPS	$\left\{\begin{array}{c}\text{D}\\\text{R}\end{array}\right\}$	1	Input	Parameter that assigns an upper limit to the absolute error for use in the eigenvalue convergence test. (See Note (d))
6	E	$\left\{\begin{array}{c}\text{D}\\\text{R}\end{array}\right\}$	M	Output	Eigenvalues
7	M	I	1	Input	The number of m of eigenvalues to be obtained.
8	VR	$\left\{\begin{array}{c}\text{D}\\\text{R}\end{array}\right\}$	LNV, M	Output	Real part (column vector) of eigenvectors corresponding to each eigenvalue
9	VI	$\left\{\begin{array}{c}\text{D}\\\text{R}\end{array}\right\}$	LNV, M	Output	Imaginary parts (column vectors) of eigenvectors corresponding to each eigenvalue
10	LNV	I	1	Input	Adjustable dimension of arrays VR and VI

No.	Argument	Type	Size	Input/ Output	Contents
11	ISW	I	1	Input	Processing switch ISW≥0: Obtain M eigenvalues from the largest one. ISW<0: Obtain M eigenvalues from the smallest one.
12	IW1	I	M	Output	Eigenvector flag (See Note (e))
13	W1	$\left\{\begin{matrix}\mathrm{D}\\\mathrm{R}\end{matrix}\right\}$	$10 \times \mathrm{N}$	Work	Work area
14	IERR	I	1	Output	Error indicator

(4) **Restrictions**

(a) $0 < \mathrm{N} \leq \mathrm{LNA}, \mathrm{LNV}$

(b) $0 < \mathrm{M} \leq \mathrm{N}$

(5) **Error indicator**

IERR value	Meaning	Processing
0	Normal termination.	
1000	N was equal to 1.	$\mathrm{E}(1) \leftarrow \mathrm{AR}(1,1)$, $\mathrm{VR}(1,1) \leftarrow 1.0$ and $\mathrm{VI}(1,1) \leftarrow 0.0$ are performed.
2000	The maximum number of iterations was exceeded by the inverse iterations for obtaining eigenvectors.	Some eigenvectors are obtained with low precision, and processing continues. (See Note (e).)
3000	Restriction (a) or (b) was not satisfied.	Processing is aborted.

(6) **Notes**

(a) The real and imaginary parts of the Hermitian matrix should be stored only in the upper triangular portions of arrays AR and AI respectively (See Appendix B).

(b) If ISW≥0, the eigenvalues are stored in descending order. If ISW<0, they are stored in ascending order.

(c) Eigenvalue calculations are appropriately divided up between the root-free QR method and Bisection method internally.

(d) If EPS ≤ 0, the optimum value is automatically set internally. Normally, a negative value should be set so that this value will be set automatically. EPS is used to obtain eigenvalues by using the Bisection method.

(e) If the maximum number of iterations is exceeded when using the inverse iteration method (IERR = 2000 is output), the following processing is performed.
If IW1(i) = 0: The i-th eigenvector calculation is normally terminated.
If IW1(i) $\neq$ 0: The convergence condition is not satisfied for the i-th eigenvector calculation, and the eigenvector precision is low. In this case, the iteration count is set for IW1(i).
If processing is normally terminated (IERR = 0 is output), IW1(i) = 0 is set.

(f) The eigenvectors are an orthonormal set.

(g) If eigenvectors are not required, use 4.7.4 $\left\{\begin{matrix}\text{ZCHRSN}\\\text{CCHRSN}\end{matrix}\right\}$.

(7) **Example**

(a) Problem

Obtain the three largest eigenvalues of the Hermitian matrix A.

$$A=\begin{bmatrix} 7 & 3 & 1+2i & -1+2i \\ 3 & 7 & 1-2i & -1-2i \\ 1-2i & 1+2i & 7 & -3 \\ -1-2i & -1+2i & -3 & 7 \end{bmatrix}$$

and their corresponding eigenvectors.

(b) Input data

Real part AR and imaginary part AI of matrix A, LNA=11, N=4, EPS=−1.0, M=3, LNV=11 and ISW=1.

(c) Main program

```
      PROGRAM ACHRSS
! *** EXAMPLE OF ZCHRSS ***
      IMPLICIT REAL(8) (A-H,O-Z)
      CHARACTER*80 FMT
      PARAMETER ( LNA = 11, LNV = 11 )
      DIMENSION AR(LNA,LNA), AI(LNA,LNA), E(LNA),&
                VR(LNV,LNV), VI(LNV,LNV), IW1(LNA), W1(10*LNA)
!
      READ(5,*) N, M
      DO 10 I=1, N
         READ(5,*) (AR(I,J), AI(I,J), J=I, N)
   10 CONTINUE
!
      WRITE(6,1000) N, M
      DO 20 I=1, N
         WRITE(FMT,1100) N
         WRITE(6,FMT) (AR(J,I), -AI(J,I), J=1, I-1),&
                      (AR(I,J),  AI(I,J), J=I, N)
   20 CONTINUE
!
      ISW = 1
      EPS = -1.0D0
!
      CALL ZCHRSS(AR,AI,LNA,N,EPS,E,M,VR,VI,LNV,ISW,IW1,W1,IERR)
!
      WRITE(6,1200) IERR
!
      DO 40 J=1, M-1, 2
         WRITE(6,1300) (’EIGENVALUE ’, I=1, 2)
         WRITE(6,1400) E(J), E(J+1)
         WRITE(6,1300) (’EIGENVECTOR’, I=1, 2)
         DO 30 I=1, N
            WRITE(6,1500) VR(I,J), VI(I,J), VR(I,J+1), VI(I,J+1)
   30    CONTINUE
   40 CONTINUE
      IF(MOD(M,2).NE.0) THEN
         WRITE(6,1300) ’EIGENVALUE ’
         WRITE(6,1400) E(M)
         WRITE(6,1300) ’EIGENVECTOR’
         DO 50 I=1, N
            WRITE(6,1500) VR(I,M), VI(I,M)
   50    CONTINUE
      ENDIF
      STOP
!
 1000 FORMAT(’ ’,/,/,&
             ’ ***  ZCHRSS  ***’,/,/,&
             ’  **  INPUT  **’,/,/,&
             ’      N = ’, I4,/,/,&
             ’      M = ’, I4,/,/,&
             ’      INPUT MATRIX A ( REAL,IMAGINARY )’,/)
 1100 FORMAT(’( ,5X,’, I2,’(’’(’’,F5.1,’’ ,’’,F5.1,’’) ’’))’)
 1200 FORMAT(’ ’,/,/,&
             ’  **  OUTPUT  **’,/,/,&
             ’      IERR = ’, I4)
 1300 FORMAT(’ ’,/, 2(14X, A11, 8X))
 1400 FORMAT(’ ’, 2(12X, 1PD14.7, 7X))
 1500 FORMAT(’ ’, 2(5X, F12.8, ’ ,’, F12.8, 2X))
      END
```

(d) Output results

```
***  ZCHRSS  ***

 **  INPUT  **

     N =    4

     M =    3

     INPUT MATRIX A ( REAL,IMAGINARY )

    (  7.0 ,  0.0) (  3.0 ,  0.0) (  1.0 ,  2.0) ( -1.0 ,  2.0)
    (  3.0 ,  0.0) (  7.0 ,  0.0) (  1.0 , -2.0) ( -1.0 , -2.0)
    (  1.0 , -2.0) (  1.0 ,  2.0) (  7.0 ,  0.0) ( -3.0 ,  0.0)
    ( -1.0 , -2.0) ( -1.0 ,  2.0) ( -3.0 ,  0.0) (  7.0 ,  0.0)

 **  OUTPUT  **

     IERR =    0

             EIGENVALUE                       EIGENVALUE
             1.2000000D+01                    8.0000000D+00

             EIGENVECTOR                      EIGENVECTOR
       0.50000000 ,  0.00000000         0.00000000 ,  0.00000000
       0.50000000 ,  0.00000000        -0.09987868 ,  0.70001732
       0.50000000 , -0.00000000        -0.30006932 , -0.39994800
      -0.50000000 , -0.00000000        -0.39994800 ,  0.30006932

             EIGENVALUE
             8.0000000D+00

             EIGENVECTOR
       0.70710678 ,  0.00000000
      -0.00000000 , -0.00000000
      -0.35355339 , -0.35355339
       0.35355339 , -0.35355339
```

4.7.4 ZCHRSN, CCHRSN Eigenvalues of a Hermitian Matrix

(1) **Function**

ZCHRSN or CCHRSN uses the Householder method, root-free QR method, or Bisection method to obtain the m largest or m smallest eigenvalues of the Hermitian matrix A=(AR, AI) (two-dimensional array type) (upper triangular type) (real argument type).

(2) **Usage**

Double precision:
CALL ZCHRSN (AR, AI, LNA, N, EPS, E, M, ISW, W1, IERR)
Single precision:
CALL CCHRSN (AR, AI, LNA, N, EPS, E, M, ISW, W1, IERR)

(3) **Arguments**

D:Double precision real Z:Double precision complex
R:Single precision real C:Single precision complex
I: { INTEGER(4) as for 32bit Integer
INTEGER(8) as for 64bit Integer }

No.	Argument	Type	Size	Input/ Output	Contents
1	AR	D R	LNA, N	Input	Real part of Hermitian matrix A (two-dimensional array type) (upper triangular type)
				Output	Input-time contents are not retained.
2	AI	D R	LNA, N	Input	Imaginary part of Hermitian matrix A (two-dimensional array type) (upper triangular type)
				Output	Input-time contents are not retained.
3	LNA	I	1	Input	Adjustable dimension of arrays AR and AI
4	N	I	1	Input	Order of matrix A
5	EPS	D R	1	Input	Parameter that assigns an upper limit to the absolute error for use in the eigenvalue convergence test. (See Note (d))
6	E	D R	M	Output	Eigenvalues
7	M	I	1	Input	The number of m of eigenvalues to be obtained.
8	ISW	I	1	Input	Processing switch ISW≥0: Obtain M eigenvalues from the largest one. ISW<0: Obtain M eigenvalues from the smallest one.
9	W1	D R	5 × N	Work	Work area
10	IERR	I	1	Output	Error indicator

(4) **Restrictions**

(a) $0 < \mathrm{N} \leq \mathrm{LNA}$

(b) $0 < \mathrm{M} \leq \mathrm{N}$

(5) **Error indicator**

IERR value	Meaning	Processing
0	Normal termination.	
1000	N was equal to 1.	$\mathrm{E}(1) \leftarrow \mathrm{AR}(1,1)$ is performed.
3000	Restriction (a) or (b) was not satisfied.	Processing is aborted.

(6) **Notes**

(a) The real and imaginary parts of the Hermitian matrix should be stored only in the upper triangular portions of arrays AR and AI respectively (See Appendix B).

(b) If ISW≥0, the eigenvalues are stored in descending order. If ISW<0, they are stored in ascending order.

(c) Eigenvalue calculations are appropriately divided up between the root-free QR method and Bisection method internally.

(d) If EPS ≤ 0, the optimum value is automatically set internally. Normally, a negative value should be set so that this value will be set automatically. EPS is used to obtain eigenvalues by using the Bisection method.

4.7.5 ZCHREE, CCHREE Eigenvalues in an Interval and Their Eigenvectors of a Hermitian Matrix (Interval Specified)

(1) **Function**

ZCHREE or CCHREE uses the Householder method and the Bisection method to obtain the m largest or m smallest eigenvalues in a specified interval of the Hermitian matrix A=(AR, AI) (two-dimensional array type) (upper triangular type) (real argument type) and the inverse iteration method to obtain the corresponding eigenvectors.

(2) **Usage**

Double precision:
CALL ZCHREE (AR, AI, LNA, N, EPS, E, M, E1, E2,VR, VI, LNV, IW1, W1, IERR)
Single precision:
CALL CCHREE (AR, AI, LNA, N, EPS, E, M, E1, E2,VR, VI, LNV, IW1, W1, IERR)

(3) **Arguments**

D:Double precision real Z:Double precision complex I: $\left\{\begin{matrix}\text{INTEGER(4) as for 32bit Integer}\\\text{INTEGER(8) as for 64bit Integer}\end{matrix}\right\}$
R:Single precision real C:Single precision complex

No.	Argument	Type	Size	Input/ Output	Contents
1	AR	$\left\{\begin{matrix}\text{D}\\\text{R}\end{matrix}\right\}$	LNA, N	Input	Real part of Hermitian matrix A (two-dimensional array type) (upper triangular type)
				Output	Input-time contents are not retained.
2	AI	$\left\{\begin{matrix}\text{D}\\\text{R}\end{matrix}\right\}$	LNA, N	Input	Imaginary part of Hermitian matrix A (two-dimensional array type) (upper triangular type)
				Output	Input-time contents are not retained.
3	LNA	I	1	Input	Adjustable dimension of arrays AR and AI
4	N	I	1	Input	Order of matrix A
5	EPS	$\left\{\begin{matrix}\text{D}\\\text{R}\end{matrix}\right\}$	1	Input	Parameter that assigns an upper limit to the absolute error for use in the eigenvalue convergence test. (See Note (b))
6	E	$\left\{\begin{matrix}\text{D}\\\text{R}\end{matrix}\right\}$	M	Output	Eigenvalues
7	M	I	1	Input	Maximum number of the eigenvalues to be computed
				Output	Number of the obtained eigenvalues

No.	Argument	Type	Size	Input/ Output	Contents
8	E1	$\begin{Bmatrix}\text{D}\\\text{R}\end{Bmatrix}$	1	Input	E1 < E2: Obtain M eigenvalues in the interval [E1, E2] from the smallest one. (E2 is upper bound.) E1>E2: Obtain M eigenvalues in the interval [E1, E2] from the largest one. (E2 is lower bound.) (See Notes (c) and (d))
9	E2	$\begin{Bmatrix}\text{D}\\\text{R}\end{Bmatrix}$	1	Input	
10	VR	$\begin{Bmatrix}\text{D}\\\text{R}\end{Bmatrix}$	LNV, M	Output	Real part (column vector) of eigenvectors corresponding to each eigenvalue
11	VI	$\begin{Bmatrix}\text{D}\\\text{R}\end{Bmatrix}$	LNV, M	Output	Imaginary parts (column vectors) of eigenvectors corresponding to each eigenvalue
12	LNV	I	1	Input	Adjustable dimension of arrays VR and VI
13	IW1	I	M	Output	Eigenvector flag (See Note (e))
14	W1	$\begin{Bmatrix}\text{D}\\\text{R}\end{Bmatrix}$	$10 \times \text{N}$	Work	Work area
15	IERR	I	1	Output	Error indicator

(4) **Restrictions**

(a) $0 < \text{N} \leq \text{LNA}, \text{LNV}$

(b) $0 < \text{M} \leq \text{N}$

(5) **Error indicator**

IERR value	Meaning	Processing
0	Normal termination.	
1000	N was equal to 1.	$\text{E}(1) \leftarrow \text{AR}(1,1)$, $\text{VR}(1,1) \leftarrow 1.0$ and $\text{VI}(1,1) \leftarrow 0.0$ are performed.
1500	The number of eigenvalues between E1 and E2 is less than M.	All the eigenvalues and the corresponding eigenvectors between E1 and E2 are obtained and the number of the found eigenvalue is output to M.
2000	The maximum number of iterations was exceeded by the inverse iterations for obtaining eigenvectors.	Some eigenvectors are obtained with low precision, and processing continues. (See Note (e).)
3000	Restriction (a) or (b) was not satisfied.	Processing is aborted.

(6) **Notes**

(a) The real and imaginary parts of the Hermitian matrix should be stored only in the upper triangular portions of arrays AR and AI respectively (See Appendix B).

(b) If EPS ≤ 0, the optimum value is automatically set internally. Normally, a negative value should be set so that this value will be set automatically. EPS is used to obtain eigenvalues by using the Bisection method.

(c) If E1 < E2 the obtained eigenvalues and eigenvectors are stored in ascending order. On the other hand, if E1 > E2 the eigenvalues and eigenvectors are stored in descending order.

(d) If E1 = E2, the eigenvalues in the interval [E1 − EPS, E1 + EPS] are obtained. Normally, E1 should be set to be different E2.

(e) If the maximum number of iterations is exceeded when using the inverse iteration method (IERR = 2000 is output), the following processing is performed.
If IW1(i) = 0: The i-th eigenvector calculation is normally terminated.
If IW1(i) $\neq$ 0: The convergence condition is not satisfied for the i-th eigenvector calculation, and the eigenvector precision is low. In this case, the iteration count is set for IW1(i).
If processing is normally terminated (IERR = 0 is output), IW1(i) = 0 is set.

(f) The eigenvectors are an orthonormal set.

(g) If eigenvectors are not required, use 4.7.6 $\left\{\begin{matrix}\text{ZCHREN}\\ \text{CCHREN}\end{matrix}\right\}$.

(7) **Example**

(a) Problem
Obtain the three eigenvalues in the interval [5, 15] from the largest one of the following Hermitian matrix A:

$$A = \begin{bmatrix} 7 & 3 & 1+2i & -1+2i \\ 3 & 7 & 1-2i & -1-2i \\ 1-2i & 1+2i & 7 & -3 \\ -1-2i & -1+2i & -3 & 7 \end{bmatrix}$$

and their corresponding eigenvectors.

(b) Input data
Real part AR and imaginary part AI of matrix A, LNA=11, N=4, EPS=−1.0, M=3, E1=15, E2=5 and LNV=11.

(c) Main program

```
      PROGRAM ACHREE
! *** EXAMPLE OF ZCHREE ***
      IMPLICIT REAL(8) (A-H,O-Z)
      CHARACTER*80 FMT
      PARAMETER ( LNA = 11, LNV = 11 )
      DIMENSION AR(LNA,LNA), AI(LNA,LNA), E(LNA),&
                VR(LNV,LNV), VI(LNV,LNV), IW1(LNA), W1(10*LNA)
!
      READ(5,*) N, M, E1, E2
      DO 10 I=1, N
         READ(5,*) (AR(I,J), AI(I,J), J=I, N)
   10 CONTINUE
!
      WRITE(6,1000) N, M, E1, E2
      DO 20 I=1, N
         WRITE(FMT,1100) N
         WRITE(6,FMT) (AR(J,I), -AI(J,I), J=1, I-1),&
                      (AR(I,J),  AI(I,J), J=I, N)
   20 CONTINUE
!
!     ISW = 1
      EPS = -1.0D0
!
      CALL ZCHREE(AR,AI,LNA,N,EPS,E,M,E1,E2,VR,VI,LNV,IW1,W1,IERR)
!
      WRITE(6,1200) IERR
!
      DO 40 J=1, M-1, 2
         WRITE(6,1300) ('EIGENVALUE ', I=1, 2)
         WRITE(6,1400) E(J), E(J+1)
         WRITE(6,1300) ('EIGENVECTOR', I=1, 2)
         DO 30 I=1, N
```

```
            WRITE(6,1500) VR(I,J), VI(I,J), VR(I,J+1), VI(I,J+1)
   30    CONTINUE
   40 CONTINUE
      IF(MOD(M,2).NE.0) THEN
         WRITE(6,1300) 'EIGENVALUE '
         WRITE(6,1400) E(M)
         WRITE(6,1300) 'EIGENVECTOR'
         DO 50 I=1, N
            WRITE(6,1500) VR(I,M), VI(I,M)
   50    CONTINUE
      ENDIF
      STOP
!
 1000 FORMAT(1X,/,/,&
             1X,'***  ZCHREE  ***',/,/,&
             1X,' **  INPUT  **',/,/,&
             1X,'N = ', I4,'  M = ', I4,/,/,&
             1X,'E1= ', 1PD14.7,'  E2= ', 1PD14.7,/,/,&
             1X,'     INPUT MATRIX A ( REAL,IMAGINARY )',/)
 1100 FORMAT('(1X,5X,', I2,'(''('',F5.1,'' ,'',F5.1,'') ''))')
 1200 FORMAT(1X,/,/,&
             1X,' **  OUTPUT  **',/,/,&
             1X,'     IERR = ', I4)
 1300 FORMAT(1X,/, 2(14X, A11, 8X))
 1400 FORMAT(1X, 2(12X, 1PD14.7, 7X))
 1500 FORMAT(1X, 2(5X, F12.8, ' ,', F12.8, 2X))
      END
```

(d) Output results

```
***  ZCHREE  ***

 **  INPUT  **

N =    4  M =    3

E1=  1.5000000D+01  E2=  5.0000000D+00

     INPUT MATRIX A ( REAL,IMAGINARY )

     (  7.0 ,  0.0) (  3.0 ,  0.0) (  1.0 ,  2.0) ( -1.0 ,  2.0)
     (  3.0 ,  0.0) (  7.0 ,  0.0) (  1.0 , -2.0) ( -1.0 , -2.0)
     (  1.0 , -2.0) (  1.0 ,  2.0) (  7.0 ,  0.0) ( -3.0 ,  0.0)
     ( -1.0 , -2.0) ( -1.0 ,  2.0) ( -3.0 ,  0.0) (  7.0 ,  0.0)


 **  OUTPUT  **

     IERR =    0

             EIGENVALUE                       EIGENVALUE
             1.2000000D+01                    8.0000000D+00

             EIGENVECTOR                      EIGENVECTOR
       0.50000000 ,  0.00000000         0.70710678 ,  0.00000000
       0.50000000 ,  0.00000000        -0.00000000 , -0.00000000
       0.50000000 , -0.00000000        -0.35355339 , -0.35355339
      -0.50000000 , -0.00000000         0.35355339 , -0.35355339

             EIGENVALUE
             8.0000000D+00

             EIGENVECTOR
       0.00000000 ,  0.00000000
      -0.09987868 ,  0.70001732
      -0.30006932 , -0.39994800
      -0.39994800 ,  0.30006932
```

4.7.6 ZCHREN, CCHREN
Eigenvalues in an Interval of a Hermitian Matrix (Interval Specified)

(1) **Function**

ZCHREN or CCHREN uses the Householder method and the Bisection method to obtain the m largest or m smallest eigenvalues in a specified interval of the Hermitian matrix $A = (\mathrm{AR}, \mathrm{AI})$ (two-dimensional array type) (upper triangular type) (real argument type).

(2) **Usage**

Double precision:

CALL ZCHREN (AR, AI, LNA, N, EPS, E, M, E1, E2, W1, IERR)

Single precision:

CALL CCHREN (AR, AI, LNA, N, EPS, E, M, E1, E2, W1, IERR)

(3) **Arguments**

D:Double precision real Z:Double precision complex I: { INTEGER(4) as for 32bit Integer
R:Single precision real C:Single precision complex INTEGER(8) as for 64bit Integer }

No.	Argument	Type	Size	Input/ Output	Contents
1	AR	D R	LNA, N	Input	Real part of Hermitian matrix A (two-dimensional array type) (upper triangular type)
				Output	Input-time contents are not retained.
2	AI	D R	LNA, N	Input	Imaginary part of Hermitian matrix A (two-dimensional array type) (upper triangular type)
				Output	Input-time contents are not retained.
3	LNA	I	1	Input	Adjustable dimension of arrays AR and AI
4	N	I	1	Input	Order of matrix A
5	EPS	D R	1	Input	Parameter that assigns an upper limit to the absolute error for use in the eigenvalue convergence test. (See Note (b))
6	E	D R	M	Output	Eigenvalues
7	M	I	1	Input	Maximum number of the eigenvalues to be computed
				Output	Number of the obtained eigenvalues
8	E1	D R	1	Input	E1 < E2: Obtain M eigenvalues in the interval [E1, E2] from the smallest one. (E2 is upper bound.)
9	E2	D R	1	Input	E1>E2: Obtain M eigenvalues in the interval [E1, E2] from the largest one. (E2 is lower bound.) (See Notes (c) and (d))
10	W1	D R	5 × N	Work	Work area
11	IERR	I	1	Output	Error indicator

(4) **Restrictions**

(a) $0 < \mathrm{N} \leq \mathrm{LNA}$

(b) $0 < \mathrm{M} \leq \mathrm{N}$

(5) **Error indicator**

IERR value	Meaning	Processing
0	Normal termination.	
1000	N was equal to 1.	$\mathrm{E}(1) \leftarrow \mathrm{AR}(1,1)$ is performed.
1500	The number of eigenvalues between E1 and E2 is less than M.	All the eigenvalues and the corresponding eigenvectors between E1 and E2 are obtained and the number of the found eigenvalue is output to M.
3000	Restriction (a) or (b) was not satisfied.	Processing is aborted.

(6) **Notes**

(a) The real and imaginary parts of the Hermitian matrix should be stored only in the upper triangular portions of arrays AR and AI respectively (See Appendix B).

(b) If $\mathrm{EPS} \leq 0$, the optimum value is automatically set internally. Normally, a negative value should be set so that this value will be set automatically. EPS is used to obtain eigenvalues by using the Bisection method.

(c) If $\mathrm{E1} < \mathrm{E2}$ the obtained eigenvalues and eigenvectors are stored in ascending order. On the other hand, if E1>E2 the eigenvalues and eigenvectors are stored in descending order.

(d) If E1=E2, the eigenvalues in the interval $[\mathrm{E1} - \mathrm{EPS}, \mathrm{E1} + \mathrm{EPS}]$ are obtained. Normally, E1 should be set to be different from E2.

4.8 HERMITIAN MATRIX (TWO-DIMENSIONAL ARRAY TYPE) (UPPER TRIANGULAR TYPE) (COMPLEX ARGUMENT TYPE)

4.8.1 ZCHEAA, CCHEAA
All Eigenvalues and All Eigenvectors of a Hermitian Matrix

(1) **Function**

ZCHEAA or CCHEAA uses the Householder method or QR method to obtain all eigenvalues of the Hermitian matrix A (two-dimensional array type) (upper triangular type) (complex argument type) and all corresponding eigenvectors.

(2) **Usage**

Double precision:
CALL ZCHEAA (A, LNA, N, E, W1, W2, IERR)
Single precision:
CALL CCHEAA (A, LNA, N, E, W1, W2, IERR)

(3) **Arguments**

D:Double precision real Z:Double precision complex I: { INTEGER(4) as for 32bit Integer
R:Single precision real C:Single precision complex INTEGER(8) as for 64bit Integer }

No.	Argument	Type	Size	Input/ Output	Contents
1	A	Z C	LNA, N	Input	Hermitian matrix A (two-dimensional array type) (upper triangular type)
				Output	Eigenvectors (column vector) corresponding to each eigenvalue
2	LNA	I	1	Input	Adjustable dimension of array A
3	N	I	1	Input	Order of matrix A
4	E	D R	N	Output	Eigenvalues
5	W1	D R	N	Work	Work area
6	W2	Z C	N	Work	Work area
7	IERR	I	1	Output	Error indicator

(4) **Restrictions**

(a) $0 < \mathrm{N} \leq \mathrm{LNA}$

(5) **Error indicator**

IERR value	Meaning	Processing
0	Normal termination.	
1000	N was equal to 1.	$\mathrm{E}(1) \leftarrow \mathrm{A}(1,1)$ and $\mathrm{A}(1,1) \leftarrow (1.0, 0.0)$ are performed.
3000	Restriction (a) was not satisfied.	Processing is aborted.
$5000 + i$	The sequence did not converge in the step where the eigenvalue is obtained. $(1 \leq i \leq \mathrm{N})$	Eigenvalues obtained by this time are entered in $\mathrm{E}(1), \cdots, \mathrm{E}(\mathrm{i}-1)$ (However, the order is irregular). No eigenvector is obtained at this time.

(6) **Notes**

(a) Only the upper triangular portion of the Hermitian matrix should be stored in array A. (See Appendix B)

(b) Eigenvalues are stored in ascending order.

(c) The eigenvectors are an orthonormal set.

(d) If eigenvectors are not required, use 4.8.2 $\left\{\begin{matrix}\text{ZCHEAN}\\\text{CCHEAN}\end{matrix}\right\}$.

(7) **Example**

(a) Problem
Obtain all eigenvalues of the matrix:

$$A = \begin{bmatrix} 7 & 3 & 1+2i & -1+2i \\ 3 & 7 & 1-2i & -1-2i \\ 1-2i & 1+2i & 7 & -3 \\ -1-2i & -1+2i & -3 & 7 \end{bmatrix}$$

and their corresponding eigenvectors.

(b) Input data
Matrix A, LNA=11 and N=4.

(c) Main program

```
      PROGRAM ACHEAA
! *** EXAMPLE OF ZCHEAA ***
      IMPLICIT REAL(8) (A-H,O-Z)
      CHARACTER*80 FMT
      COMPLEX(8) A,W2
      PARAMETER ( LNA = 11 )
      DIMENSION A(LNA,LNA), E(LNA), W1(LNA), W2(LNA)
!
      READ(5,*) N
      DO 10 I=1, N
         READ(5,*) (A(I,J), J=I, N)
   10 CONTINUE
!
      WRITE(6,1000) N
      DO 20 I=1, N
         WRITE(FMT,1100) N
         WRITE(6,FMT) (DCONJG(A(J,I)), J=1, I-1), (A(I,J), J=I, N)
   20 CONTINUE
!
      CALL ZCHEAA(A,LNA,N,E,W1,W2,IERR)
!
      WRITE(6,1200) IERR
```

```
!
      DO 40 J=1, N-1, 2
         WRITE(6,1300) ('EIGENVALUE ', I=1, 2)
         WRITE(6,1400) E(J), E(J+1)
         WRITE(6,1300) ('EIGENVECTOR', I=1, 2)
         DO 30 I=1, N
            WRITE(6,1500) A(I,J), A(I,J+1)
   30    CONTINUE
   40 CONTINUE
      IF(MOD(N,2).NE.0) THEN
         WRITE(6,1300) 'EIGENVALUE '
         WRITE(6,1400) E(N)
         WRITE(6,1300) 'EIGENVECTOR'
         DO 50 I=1, N
            WRITE(6,1500) A(I,N)
   50    CONTINUE
      ENDIF
      STOP
!
 1000 FORMAT(' ',/,/,&
             ' ***  ZCHEAA  ***',/,/,&
             '  **  INPUT  **',/,/,&
             '      N = ', I4,/,/,&
             '      INPUT MATRIX A ( REAL,IMAGINARY )',/)
 1100 FORMAT('( ,5X,', I2,'(''('',F5.1,'' ,'',F5.1,'') ''))')
 1200 FORMAT(' ',/,/,&
             '  **  OUTPUT  **',/,/,&
             '      IERR = ', I4)
 1300 FORMAT(' ',/, 2(14X, A11, 8X))
 1400 FORMAT(' ', 2(12X, 1PD14.7, 7X))
 1500 FORMAT(' ', 2(5X, F12.8, ' ,', F12.8, 2X))
      END
```

(d) Output results

```
***  ZCHEAA  ***

 **  INPUT  **

     N =    4

     INPUT MATRIX A ( REAL,IMAGINARY )

    (  7.0 ,  0.0) (  3.0 ,  0.0) (  1.0 ,  2.0) ( -1.0 ,  2.0)
    (  3.0 ,  0.0) (  7.0 ,  0.0) (  1.0 , -2.0) ( -1.0 , -2.0)
    (  1.0 , -2.0) (  1.0 ,  2.0) (  7.0 ,  0.0) ( -3.0 ,  0.0)
    ( -1.0 , -2.0) ( -1.0 ,  2.0) ( -3.0 ,  0.0) (  7.0 ,  0.0)


 **  OUTPUT  **

     IERR =    0

             EIGENVALUE                       EIGENVALUE
             0.0000000D+00                    8.0000000D+00

             EIGENVECTOR                      EIGENVECTOR
       0.50000000 ,  0.00000000        -0.70710678 , -0.00000000
      -0.50000000 ,  0.00000000         0.00000000 , -0.00000000
       0.00000000 ,  0.50000000         0.35355339 ,  0.35355339
      -0.00000000 ,  0.50000000        -0.35355339 ,  0.35355339

             EIGENVALUE                       EIGENVALUE
             8.0000000D+00                    1.2000000D+01

             EIGENVECTOR                      EIGENVECTOR
       0.00000000 ,  0.00000000         0.50000000 ,  0.00000000
      -0.09987868 ,  0.70001732         0.50000000 ,  0.00000000
      -0.30006932 , -0.39994800         0.50000000 , -0.00000000
      -0.39994800 ,  0.30006932        -0.50000000 , -0.00000000
```

4.8.2 ZCHEAN, CCHEAN All Eigenvalues of a Hermitian Matrix

(1) **Function**

ZCHEAN or CCHEAN uses the Householder method or root-free QR method to obtain all eigenvalues of the Hermitian matrix A (two-dimensional array type) (upper triangular type) (complex argument type).

(2) **Usage**

Double precision:

CALL ZCHEAN (A, LNA, N, E, W1, W2, IERR)

Single precision:

CALL CCHEAN (A, LNA, N, E, W1, W2, IERR)

(3) **Arguments**

D:Double precision real Z:Double precision complex
R:Single precision real C:Single precision complex

$\text{I:} \begin{Bmatrix} \text{INTEGER(4) as for 32bit Integer} \\ \text{INTEGER(8) as for 64bit Integer} \end{Bmatrix}$

No.	Argument	Type	Size	Input/ Output	Contents
1	A	$\begin{Bmatrix} Z \\ C \end{Bmatrix}$	LNA, N	Input	Hermitian matrix A (two-dimensional array type) (upper triangular type)
				Output	Input-time contents are not retained.
2	LNA	I	1	Input	Adjustable dimension of array A
3	N	I	1	Input	Order of matrix A
4	E	$\begin{Bmatrix} D \\ R \end{Bmatrix}$	N	Output	Eigenvalues
5	W1	$\begin{Bmatrix} D \\ R \end{Bmatrix}$	N	Work	Work area
6	W2	$\begin{Bmatrix} Z \\ C \end{Bmatrix}$	N	Work	Work area
7	IERR	I	1	Output	Error indicator

(4) **Restrictions**

(a) $0 < \text{N} \leq \text{LNA}$

(5) **Error indicator**

IERR value	Meaning	Processing
0	Normal termination.	
1000	N was equal to 1.	E(1) ← A(1, 1) is performed.
3000	Restriction (a) was not satisfied.	Processing is aborted.
$5000 + i$	The sequence did not converge in the step where the eigenvalue is obtained. $(1 \leq i \leq AGN)$	Eigenvalues obtained by this time are entered in E(1), $\cdots$, E(i − 1) (However, the order is irregular).

(6) **Notes**

(a) Only the upper triangular portion of the Hermitian matrix should be stored in array A. (See Appendix B)

(b) Eigenvalues are stored in ascending order.

4.8.3 ZCHESS, CCHESS
Eigenvalues and Eigenvectors of a Hermitian Matrix

(1) **Function**

ZCHESS or CCHESS uses the Householder method, root-free QR method, or Bisection method to obtain the m largest or m smallest eigenvalues of the Hermitian matrix A (two-dimensional array type) (upper triangular type) (complex argument type) and the inverse iterative method to obtain the corresponding eigenvectors.

(2) **Usage**

Double precision:
CALL ZCHESS (A, LNA, N, EPS, E, M, VE, LNV, ISW, IW1, W1, W2, IERR)
Single precision:
CALL CCHESS (A, LNA, N, EPS, E, M, VE, LNV, ISW, IW1, W1, W2, IERR)

(3) **Arguments**

D:Double precision real   Z:Double precision complex
R:Single precision real   C:Single precision complex
I: $\left\{\begin{matrix}\text{INTEGER(4) as for 32bit Integer}\\\text{INTEGER(8) as for 64bit Integer}\end{matrix}\right\}$

No.	Argument	Type	Size	Input/ Output	Contents
1	A	$\left\{\begin{matrix}Z\\C\end{matrix}\right\}$	LNA, N	Input	Hermitian matrix A (two-dimensional array type) (upper triangular type)
				Output	Input-time contents are not retained.
2	LNA	I	1	Input	Adjustable dimension of array A.
3	N	I	1	Input	Order of matrix A
4	EPS	$\left\{\begin{matrix}D\\R\end{matrix}\right\}$	1	Input	Parameter that assigns an upper limit to the absolute error for use in the eigenvalue convergence test. (See Note (d))
5	E	$\left\{\begin{matrix}D\\R\end{matrix}\right\}$	M	Output	Eigenvalues
6	M	I	1	Input	The number of m of eigenvalues to be obtained.
7	VE	$\left\{\begin{matrix}Z\\C\end{matrix}\right\}$	LNV, M	Output	Eigenvectors (column vector) corresponding to each eigenvalue
8	LNV	I	1	Input	Adjustable dimension of array VE
9	ISW	I	1	Input	Processing switch ISW≥0: Obtain M eigenvalues from the largest one. ISW<0: Obtain M eigenvalues from the smallest one.
10	IW1	I	M	Output	Eigenvector flag (See Note (e))

No.	Argument	Type	Size	Input/ Output	Contents
11	W1	$\begin{Bmatrix} \mathrm{D} \\ \mathrm{R} \end{Bmatrix}$	8 × N	Work	Work area
12	W2	$\begin{Bmatrix} \mathrm{Z} \\ \mathrm{C} \end{Bmatrix}$	N	Work	Work area
13	IERR	I	1	Output	Error indicator

(4) **Restrictions**

(a) $0 < \mathrm{N} \leq \mathrm{LNA}, \mathrm{LNV}$

(b) $0 < \mathrm{M} \leq \mathrm{N}$

(5) **Error indicator**

IERR value	Meaning	Processing
0	Normal termination.	
1000	N was equal to 1.	$\mathrm{E}(1) \leftarrow \mathrm{A}(1,1)$ and $\mathrm{VE}(1,1) \leftarrow (1.0, 0.0)$ are performed.
2000	The maximum number of iterations was exceeded by the inverse iterations for obtaining eigenvectors.	Some eigenvectors are obtained with low precision, and processing continues. (See Note (e).)
3000	Restriction (a) or (b) was not satisfied.	Processing is aborted.

(6) **Notes**

(a) Only the upper triangular portion of the Hermitian matrix should be stored in array A. (See Appendix B)

(b) If ISW≥0, the eigenvalues are stored in descending order. If ISW < 0, they are stored in ascending order.

(c) Eigenvalue calculations are appropriately divided up between the root-free QR method and Bisection method internally.

(d) If EPS ≤ 0, the optimum value is automatically set internally. Normally, a negative value should be set so that this value will be set automatically. EPS is used to obtain eigenvalues by using the Bisection method.

(e) If the maximum number of iterations is exceeded when using the inverse iteration method (IERR = 2000 is output), the following processing is performed.

If IW1(i) = 0: The i-th eigenvector calculation is normally terminated.

If IW1(i) $\neq$ 0: The convergence condition is not satisfied for the i-th eigenvector calculation, and the eigenvector precision is low. In this case, the iteration count is set for IW1(i).

If processing is normally terminated (IERR = 0 is output), IW1(i) = 0 is set.

(f) The eigenvectors are an orthonormal set.

(g) If eigenvectors are not required, use 4.8.4 $\begin{Bmatrix} \mathrm{ZCHESN} \\ \mathrm{CCHESN} \end{Bmatrix}$.

(7) **Example**

(a) Problem

Obtain the three largest eigenvalues of the following Hermitian matrix A:

$$A = \begin{bmatrix} 7 & 3 & 1+2i & -1+2i \\ 3 & 7 & 1-2i & -1-2i \\ 1-2i & 1+2i & 7 & -3 \\ -1-2i & -1+2i & -3 & 7 \end{bmatrix}$$

and their corresponding eigenvectors.

(b) Input data

Matrix A, LNA=11, N=4, EPS=−1.0, M=3, LNV=11 and ISW=1.

(c) Main program

```
      PROGRAM ACHESS
! *** EXAMPLE OF ZCHESS ***
      IMPLICIT REAL(8) (A-H,O-Z)
      CHARACTER*80 FMT
      COMPLEX(8) A,VE,W2
      PARAMETER ( LNA = 11, LNV = 11 )
      DIMENSION A(LNA,LNA), E(LNA), VE(LNV,LNV),&
                IW1(LNA), W1(8*LNA), W2(LNA)
!
      READ(5,*) N, M
      DO 10 I=1, N
         READ(5,*) (A(I,J), J=I, N)
   10 CONTINUE
!
      WRITE(6,1000) N, M
      DO 20 I=1, N
         WRITE(FMT,1100) N
         WRITE(6,FMT) (DCONJG(A(J,I)), J=1, I-1), (A(I,J), J=I, N)
   20 CONTINUE
!
      ISW = 1
      EPS = -1.0D0
!
      CALL ZCHESS(A,LNA,N,EPS,E,M,VE,LNV,ISW,IW1,W1,W2,IERR)
!
      WRITE(6,1200) IERR
!
      DO 40 J=1, M-1, 2
         WRITE(6,1300) (’EIGENVALUE ’, I=1, 2)
         WRITE(6,1400) E(J), E(J+1)
         WRITE(6,1300) (’EIGENVECTOR’, I=1, 2)
         DO 30 I=1, N
            WRITE(6,1500) VE(I,J), VE(I,J+1)
   30    CONTINUE
   40 CONTINUE
      IF(MOD(M,2).NE.0) THEN
         WRITE(6,1300) ’EIGENVALUE ’
         WRITE(6,1400) E(M)
         WRITE(6,1300) ’EIGENVECTOR’
         DO 50 I=1, N
            WRITE(6,1500) VE(I,M)
   50    CONTINUE
      ENDIF
      STOP
!
 1000 FORMAT(’ ’,/,/,&
             ’ ***  ZCHESS  ***’,/,/,&
             ’  **  INPUT  **’,/,/,&
             ’      N = ’, I4,/,/,&
             ’      M = ’, I4,/,/,&
             ’      INPUT MATRIX A ( REAL,IMAGINARY )’,/)
 1100 FORMAT(’( ,5X,’, I2,’(’’(’’,F5.1,’’ ,’’,F5.1,’’) ’’))’)
 1200 FORMAT(’ ’,/,/,&
             ’  **  OUTPUT  **’,/,/,&
             ’      IERR = ’, I4)
 1300 FORMAT(’ ’,/, 2(14X, A11, 8X))
 1400 FORMAT(’ ’, 2(12X, 1PD14.7, 7X))
 1500 FORMAT(’ ’, 2(5X, F12.8, ’ ,’, F12.8, 2X))
      END
```

(d) Output results

```
***  ZCHESS  ***

 **  INPUT  **

     N =    4
```

```
    M =    3

    INPUT MATRIX A ( REAL,IMAGINARY )

   (  7.0 ,  0.0) (  3.0 ,  0.0) (  1.0 ,  2.0) ( -1.0 ,  2.0)
   (  3.0 ,  0.0) (  7.0 ,  0.0) (  1.0 , -2.0) ( -1.0 , -2.0)
   (  1.0 , -2.0) (  1.0 ,  2.0) (  7.0 ,  0.0) ( -3.0 ,  0.0)
   ( -1.0 , -2.0) ( -1.0 ,  2.0) ( -3.0 ,  0.0) (  7.0 ,  0.0)

**  OUTPUT  **

    IERR =    0

            EIGENVALUE                        EIGENVALUE
            1.2000000D+01                     8.0000000D+00

            EIGENVECTOR                       EIGENVECTOR
      0.50000000 ,  0.00000000          0.00000000 ,  0.00000000
      0.50000000 ,  0.00000000         -0.09987868 ,  0.70001732
      0.50000000 , -0.00000000         -0.30006932 , -0.39994800
     -0.50000000 , -0.00000000         -0.39994800 ,  0.30006932

            EIGENVALUE
            8.0000000D+00

            EIGENVECTOR
      0.70710678 ,  0.00000000
     -0.00000000 ,  0.00000000
     -0.35355339 , -0.35355339
      0.35355339 , -0.35355339
```

4.8.4 ZCHESN, CCHESN Eigenvalues of a Hermitian Matrix

(1) **Function**

ZCHESN or CCHESN uses the Householder method, root-free QR method, or Bisection method to obtain the m largest or m smallest eigenvalues of the Hermitian matrix A (two-dimensional array type) (upper triangular type) (complex argument type).

(2) **Usage**

Double precision:
CALL ZCHESN (A, LNA, N, EPS, E, M, ISW, W1, W2, IERR)
Single precision:
CALL CCHESN (A, LNA, N, EPS, E, M, ISW, W1, W2, IERR)

(3) **Arguments**

D:Double precision real Z:Double precision complex I: { INTEGER(4) as for 32bit Integer
R:Single precision real C:Single precision complex INTEGER(8) as for 64bit Integer }

No.	Argument	Type	Size	Input/ Output	Contents
1	A	Z C	LNA, N	Input	Hermitian matrix A (two-dimensional array type) (upper triangular type)
				Output	Input-time contents are not retained.
2	LNA	I	1	Input	Adjustable dimension of array A.
3	N	I	1	Input	Order of matrix A
4	EPS	D R	1	Input	Parameter that assigns an upper limit to the absolute error for use in the eigenvalue convergence test. (See Note (d))
5	E	D R	M	Output	Eigenvalues
6	M	I	1	Input	The number of m of eigenvalues to be obtained.
7	ISW	I	1	Input	Processing switch ISW≥0: Obtain M eigenvalues from the largest one. ISW<0: Obtain M eigenvalues from the smallest one.
8	W1	D R	2 × N	Work	Work area
9	W2	Z C	N	Work	Work area
10	IERR	I	1	Output	Error indicator

(4) **Restrictions**

(a) $0 < \mathrm{N} \leq \mathrm{LNA}$

(b) $0 < \mathrm{M} \leq \mathrm{N}$

(5) **Error indicator**

IERR value	Meaning	Processing
0	Normal termination.	
1000	N was equal to 1.	$\mathrm{E}(1) \leftarrow \mathrm{A}(1,1)$ is performed.
3000	Restriction (a) or (b) was not satisfied.	Processing is aborted.

(6) **Notes**

(a) Only the upper triangular portion of the Hermitian matrix should be stored in array A. (See Appendix B)

(b) If ISW≥0, the eigenvalues are stored in descending order. If ISW<0, they are stored in ascending order.

(c) Eigenvalue calculations are appropriately divided up between the root-free QR method and Bisection method internally.

(d) If EPS ≤ 0, the optimum value is automatically set internally. Normally, a negative value should be set so that this value will be set automatically. EPS is used to obtain eigenvalues by using the Bisection method.

4.8.5 ZCHEEE, CCHEEE Eigenvalues in an Interval and their Eigenvectors of a Hermitian Matrix (Interval Specified)

(1) **Function**

ZCHEEE or CCHEEE uses the Householder method and the Bisection method to obtain the m largest or m smallest eigenvalues in a specified interval of the Hermitian matrix A (two-dimensional array type) (upper triangular type) (complex argument type) and the inverse iterative method to obtain the corresponding eigenvectors.

(2) **Usage**

Double precision:

CALL ZCHEEE (A, LNA, N, EPS, E, M, E1, E2, VE, LNV, IW1, W1, W2, IERR)

Single precision:

CALL CCHEEE (A, LNA, N, EPS, E, M, E1, E2, VE, LNV, IW1, W1, W2, IERR)

(3) **Arguments**

D:Double precision real Z:Double precision complex I: $\left\{\begin{array}{l}\text{INTEGER(4) as for 32bit Integer}\\ \text{INTEGER(8) as for 64bit Integer}\end{array}\right\}$
R:Single precision real C:Single precision complex

No.	Argument	Type	Size	Input/ Output	Contents
1	A	$\left\{\begin{matrix}\text{Z}\\ \text{C}\end{matrix}\right\}$	LNA, N	Input	Hermitian matrix A (two-dimensional array type) (upper triangular type)
				Output	Input-time contents are not retained.
2	LNA	I	1	Input	Adjustable dimension of array A.
3	N	I	1	Input	Order of matrix A
4	EPS	$\left\{\begin{matrix}\text{D}\\ \text{R}\end{matrix}\right\}$	1	Input	Parameter that assigns an upper limit to the absolute error for use in the eigenvalue convergence test. (See Note (b))
5	E	$\left\{\begin{matrix}\text{D}\\ \text{R}\end{matrix}\right\}$	M	Output	Eigenvalues
6	M	I	1	Input	Maximum number of the eigenvalues to be computed
				Output	Number of the obtained eigenvalues
7	E1	$\left\{\begin{matrix}\text{D}\\ \text{R}\end{matrix}\right\}$	1	Input	E1 < E2: Obtain M eigenvalues in the interval [E1, E2] from the smallest one. (E2 is upper bound.) E1 > E2: Obtain M eigenvalues in the interval [E1, E2] from the largest one. (E2 is lower bound.) (See Notes (c) and (d))
8	E2	$\left\{\begin{matrix}\text{D}\\ \text{R}\end{matrix}\right\}$	1	Input	

No.	Argument	Type	Size	Input/ Output	Contents
9	VE	$\begin{Bmatrix} \mathrm{Z} \\ \mathrm{C} \end{Bmatrix}$	LNV, M	Output	Eigenvectors (column vector) corresponding to each eigenvalue
10	LNV	I	1	Input	Adjustable dimension of array VE
11	IW1	I	M	Output	Eigenvector flag (See Note (e))
12	W1	$\begin{Bmatrix} \mathrm{D} \\ \mathrm{R} \end{Bmatrix}$	$8 \times \mathrm{N}$	Work	Work area
13	W2	$\begin{Bmatrix} \mathrm{Z} \\ \mathrm{C} \end{Bmatrix}$	N	Work	Work area
14	IERR	I	1	Output	Error indicator

(4) **Restrictions**

(a) $0 < \mathrm{N} \leq \mathrm{LNA}, \mathrm{LNV}$

(b) $0 < \mathrm{M} \leq \mathrm{N}$

(5) **Error indicator**

IERR value	Meaning	Processing
0	Normal termination.	
1000	N was equal to 1.	$\mathrm{E}(1) \leftarrow \mathrm{A}(1,1)$ and $\mathrm{VE}(1,1) \leftarrow (1.0, 0.0)$ are performed.
1500	The number of eigenvalues between E1 and E2 is less than M.	All the eigenvalues and the corresponding eigenvectors between E1 and E2 are obtained and the number of the found eigenvalue is output to M.
2000	The maximum number of iterations was exceeded by the inverse iterations for obtaining eigenvectors.	Some eigenvectors are obtained with low precision, and processing continues. (See Note (e).)
3000	Restriction (a) or (b) was not satisfied.	Processing is aborted.

(6) **Notes**

(a) Only the upper triangular portion of the Hermitian matrix should be stored in array A. (See Appendix B)

(b) If $\mathrm{EPS} \leq 0$, the optimum value is automatically set internally. Normally, a negative value should be set so that this value will be set automatically. EPS is used to obtain eigenvalues by using the Bisection method.

(c) If $\mathrm{E1} < \mathrm{E2}$ the obtained eigenvalues and eigenvectors are stored in ascending order. On the other hand, if E1>E2 the eigenvalues and eigenvectors are stored in descending order.

(d) If E1=E2, the eigenvalues in the interval [$\mathrm{E1} - \mathrm{EPS}$, $\mathrm{E1} + \mathrm{EPS}$] are obtained. Normally, E1 should be set to be different E2.

(e) If the maximum number of iterations is exceeded when using the inverse iteration method (IERR = 2000 is output), the following processing is performed.

If IW1(i) = 0: The i-th eigenvector calculation is normally terminated.

If IW1(i) $\neq$ 0: The convergence condition is not satisfied for the i-th eigenvector calculation, and the eigenvector precision is low. In this case, the iteration count is set for IW1(i).

If processing is normally terminated (IERR = 0 is output), IW1(i) = 0 is set.

(f) The eigenvectors are an orthonormal set.

(g) If eigenvectors are not required, use 4.8.6 $\left\{\begin{array}{c}\text{ZCHEEN}\\ \text{CCHEEN}\end{array}\right\}$.

(7) **Example**

(a) Problem

Obtain the three eigenvalues in the interval [15, 5] from the largest one of the following Hermitian matrix A:

$$A = \left[\begin{array}{llll} 7 & 3 & 1+2i & -1+2i \\ 3 & 7 & 1-2i & -1-2i \\ 1-2i & 1+2i & \;\;7 & \;\;-3 \\ -1-2i & -1+2i & -3 & \;\;\;\;7 \end{array}\right]$$

and their corresponding eigenvectors.

(b) Input data

Matrix A, LNA=11, N=4, EPS=−1.0, M=3, E1=15, E2=5 and LNV=11.

(c) Main program

```
      PROGRAM ACHEEE
! *** EXAMPLE OF ZCHEEE ***
      IMPLICIT REAL(8) (A-H,O-Z)
      CHARACTER*80 FMT
      COMPLEX(8) A,VE,W2
      PARAMETER ( LNA = 11, LNV = 11 )
      DIMENSION A(LNA,LNA), E(LNA), VE(LNV,LNV),&
                IW1(LNA), W1(8*LNA), W2(LNA)
!
      READ(5,*) N, M, E1, E2
      DO 10 I=1, N
         READ(5,*) (A(I,J), J=I, N)
   10 CONTINUE
!
      WRITE(6,1000) N, M, E1, E2
      DO 20 I=1, N
         WRITE(FMT,1100) N
         WRITE(6,FMT) (DCONJG(A(J,I)), J=1, I-1), (A(I,J), J=I, N)
   20 CONTINUE
!
!     ISW = 1
      EPS = -1.0D0
!
      CALL ZCHEEE(A,LNA,N,EPS,E,M,E1,E2,VE,LNV,IW1,W1,W2,IERR)
!
      WRITE(6,1200) IERR
!
      DO 40 J=1, M-1, 2
         WRITE(6,1300) ('EIGENVALUE ', I=1, 2)
         WRITE(6,1400) E(J), E(J+1)
         WRITE(6,1300) ('EIGENVECTOR', I=1, 2)
         DO 30 I=1, N
            WRITE(6,1500) VE(I,J), VE(I,J+1)
   30    CONTINUE
   40 CONTINUE
      IF(MOD(M,2).NE.0) THEN
         WRITE(6,1300) 'EIGENVALUE '
         WRITE(6,1400) E(M)
         WRITE(6,1300) 'EIGENVECTOR'
         DO 50 I=1, N
            WRITE(6,1500) VE(I,M)
   50    CONTINUE
      ENDIF
      STOP
!
```

```
 1000 FORMAT(1X,/,/,&
             1X,'***  ZCHEEE  ***',/,/,&
             1X,' **  INPUT  **',/,/,&
             1X,'N = ', I4,'  M = ', I4,/,/,&
             1X,'E1= ', 1PD14.7,'  E2= ', 1PD14.7,/,/,&
             1X,'     INPUT MATRIX A ( REAL,IMAGINARY )',/)
 1100 FORMAT('(1X,5X,', I2,'(''('',F5.1,'' ,'',F5.1,'') ''))')
 1200 FORMAT(1X,/,/,&
             1X,' **  OUTPUT  **',/,/,&
             1X,'     IERR = ', I4)
 1300 FORMAT(1X,/, 2(14X, A11, 8X))
 1400 FORMAT(1X, 2(12X, 1PD14.7, 7X))
 1500 FORMAT(1X, 2(5X, F12.8, ' ,', F12.8, 2X))
      END
```

(d) Output results

```
***  ZCHEEE  ***

 **  INPUT  **

N =    4  M =    3

E1=  1.5000000D+01  E2=  5.0000000D+00

     INPUT MATRIX A ( REAL,IMAGINARY )

     (  7.0 ,  0.0) (  3.0 ,  0.0) (  1.0 ,  2.0) ( -1.0 ,  2.0)
     (  3.0 ,  0.0) (  7.0 ,  0.0) (  1.0 , -2.0) ( -1.0 , -2.0)
     (  1.0 , -2.0) (  1.0 ,  2.0) (  7.0 ,  0.0) ( -3.0 ,  0.0)
     ( -1.0 , -2.0) ( -1.0 ,  2.0) ( -3.0 ,  0.0) (  7.0 ,  0.0)


 **  OUTPUT  **

     IERR =    0

             EIGENVALUE                       EIGENVALUE
             1.2000000D+01                    8.0000000D+00

             EIGENVECTOR                      EIGENVECTOR
       0.50000000 ,   0.00000000         0.70710678 ,   0.00000000
       0.50000000 ,   0.00000000        -0.00000000 ,   0.00000000
       0.50000000 ,  -0.00000000        -0.35355339 ,  -0.35355339
      -0.50000000 ,  -0.00000000         0.35355339 ,  -0.35355339

             EIGENVALUE
             8.0000000D+00

             EIGENVECTOR
       0.00000000 ,   0.00000000
      -0.09987868 ,   0.70001732
      -0.30006932 ,  -0.39994800
      -0.39994800 ,   0.30006932
```

4.8.6 ZCHEEN, CCHEEN Eigenvalues in an Interval of a Hermitian Matrix (Interval Specified)

(1) **Function**

ZCHEEN or CCHEEN uses the Householder method and the Bisection method to obtain the m largest or m smallest eigenvalues of the Hermitian matrix A (two-dimensional array type) (upper triangular type) (complex argument type).

(2) **Usage**

Double precision:

```
  CALL ZCHEEN  (A, LNA, N, EPS, E, M, E1, E2, W1, W2, IERR)
```

Single precision:

```
  CALL CCHEEN  (A, LNA, N, EPS, E, M, E1, E2, W1, W2, IERR)
```

(3) **Arguments**

D:Double precision real Z:Double precision complex I: $\left\{\begin{matrix}\text{INTEGER(4) as for 32bit Integer}\\ \text{INTEGER(8) as for 64bit Integer}\end{matrix}\right\}$
R:Single precision real C:Single precision complex

No.	Argument	Type	Size	Input/ Output	Contents
1	A	$\begin{Bmatrix}Z\\C\end{Bmatrix}$	LNA, N	Input	Hermitian matrix A (two-dimensional array type) (upper triangular type)
				Output	Input-time contents are not retained.
2	LNA	I	1	Input	Adjustable dimension of array A.
3	N	I	1	Input	Order of matrix A
4	EPS	$\begin{Bmatrix}D\\R\end{Bmatrix}$	1	Input	Parameter that assigns an upper limit to the absolute error for use in the eigenvalue convergence test. (See Note (b))
5	E	$\begin{Bmatrix}D\\R\end{Bmatrix}$	M	Output	Eigenvalues
6	M	I	1	Input	Maximum number of the eigenvalues to be computed
				Output	Number of the obtained eigenvalues
7	E1	$\begin{Bmatrix}D\\R\end{Bmatrix}$	1	Input	E1<E2: Obtain M eigenvalues in the interval [E1, E2] from the smallest one. (E2 is upper bound.) E1>E2: Obtain M eigenvalues in the interval [E1, E2] from the largest one. (E2 is lower bound.) (See Notes (c) and (d))
8	E2	$\begin{Bmatrix}D\\R\end{Bmatrix}$	1	Input	

No.	Argument	Type	Size	Input/ Output	Contents
9	W1	$\begin{Bmatrix} \text{D} \\ \text{R} \end{Bmatrix}$	$2 \times \text{N}$	Work	Work area
10	W2	$\begin{Bmatrix} \text{Z} \\ \text{C} \end{Bmatrix}$	N	Work	Work area
11	IERR	I	1	Output	Error indicator

(4) **Restrictions**

(a) $0 < \text{N} \leq \text{LNA}$

(b) $0 < \text{M} \leq \text{N}$

(5) **Error indicator**

IERR value	Meaning	Processing
0	Normal termination.	
1000	N was equal to 1.	$\text{E}(1) \leftarrow \text{A}(1,1)$ is performed.
1500	The number of eigenvalues between E1 and E2 is less than M.	All the eigenvalues and the corresponding eigenvectors between E1 and E2 are obtained and the number of the found eigenvalue is output to M.
3000	Restriction (a) or (b) was not satisfied.	Processing is aborted.

(6) **Notes**

(a) Only the upper triangular portion of the Hermitian matrix should be stored in array A. (See Appendix B)

(b) If EPS ≤ 0, the optimum value is automatically set internally. Normally, a negative value should be set so that this value will be set automatically. EPS is used to obtain eigenvalues by using the Bisection method.

(c) If E1 $<$ E2 the obtained eigenvalues and eigenvectors are stored in ascending order. On the other hand, if E1>E2 the eigenvalues and eigenvectors are stored in descending order.

(d) If E1=E2, the eigenvalues in the interval [E1 $-$ EPS, E1 $+$ EPS] are obtained. Normally, E1 should be set to be different E2.

4.9 REAL SYMMETRIC BAND MATRIX (SYMMETRIC BAND TYPE)

4.9.1 DCSBAA, RCSBAA
All Eigenvalues and All Eigenvectors of a Real Symmetric Band Matrix

(1) **Function**

DCSBAA or RCSBAA uses the Householder method and QR method to obtain all eigenvalues of the real symmetric band matrix A (symmetric band type) and all corresponding eigenvectors.

(2) **Usage**

Double precision:
CALL DCSBAA (A, LMA, N, MB, E, VE, LNV, W1, IERR)
Single precision:
CALL RCSBAA (A, LMA, N, MB, E, VE, LNV, W1, IERR)

(3) **Arguments**

D:Double precision real Z:Double precision complex I: $\begin{Bmatrix} \text{INTEGER(4) as for 32bit Integer} \\ \text{INTEGER(8) as for 64bit Integer} \end{Bmatrix}$
R:Single precision real C:Single precision complex

No.	Argument	Type	Size	Input/ Output	Contents
1	A	$\begin{Bmatrix} D \\ R \end{Bmatrix}$	LMA, N	Input	Real symmetric band matrix A (symmetric band type) (See Note (a)).
				Output	Input-time contents are not retained.
2	LMA	I	1	Input	Adjustable dimension of array A.
3	N	I	1	Input	Order of matrix A.
4	MB	I	1	Input	Band width of matrix A.
5	E	$\begin{Bmatrix} D \\ R \end{Bmatrix}$	N	Output	Eigenvalues.
6	VE	$\begin{Bmatrix} D \\ R \end{Bmatrix}$	LNV, N	Output	Eigenvectors (column vector) corresponding to each eigenvalue.
7	LNV	I	1	Input	Adjustable dimension of array VE.
8	W1	$\begin{Bmatrix} D \\ R \end{Bmatrix}$	N	Work	Work area
9	IERR	I	1	Output	Error indicator

(4) **Restrictions**

(a) $0 < \mathrm{N} \le \mathrm{LNV}$

(b) $0 \le \mathrm{MB} < \mathrm{N}$

(c) $\mathrm{MB} < \mathrm{LMA}$

(5) **Error indicator**

IERR value	Meaning	Processing
0	Normal termination.	
1000	N was equal to 1.	E(1) ← A(1, 1) and VE(1, 1) ← 1.0 are performed.
1100	MB was equal to 0.	E(i) ← A(1, i) $(1 \le i \le \text{N})$ and VE ← I (I is unit matrix) are performed.
3000	Restriction (a), (b) or (c) was not satisfied.	Processing is aborted.
$5000 + i$	The sequence did not converge in the step where the eigenvalue is obtained. $(1 \le i \le \text{N})$	Eigenvalues obtained by this time are entered in $\text{E}(1), \cdots, \text{E}(\text{i}-1)$ and eigenvectors corresponding to them are entered in A (However, the order is irregular).

(6) **Notes**

(a) The real symmetric band matrix is compressed into symmetric band type having (MB+1) rows and N columns and stored in array A (See Appendix B).

(b) Eigenvalues are stored in ascending order.

(c) The eigenvectors are an orthonormal set.

(d) If eigenvectors are not required, use 4.9.2 $\left\{\begin{matrix}\text{DCSBAN}\\ \text{RCSBAN}\end{matrix}\right\}$.

(7) **Example**

(a) Problem

Obtain all eigenvalues of the matrix:

$$A = \begin{bmatrix} 5 & -4 & 1 & & & & \\ -4 & 6 & -4 & 1 & & 0 & \\ 1 & -4 & 6 & -4 & 1 & & \\ & 1 & -4 & 6 & -4 & 1 & \\ & & 1 & -4 & 6 & -4 & 1 \\ & 0 & & 1 & -4 & 6 & -4 \\ & & & & 1 & -4 & 5 \end{bmatrix}$$

and their corresponding eigenvectors.

(b) Input data

Matrix A, LMA=11, N=7, MB=2 and LNV=11.

(c) Main program

```
      PROGRAM BCSBAA
! *** EXAMPLE OF DCSBAA ***
      IMPLICIT REAL(8) (A-H,O-Z)
      CHARACTER*80 FMT
      PARAMETER ( LMA = 11, LNV = 11 )
      DIMENSION A(LMA,LMA), E(LMA), VE(LNV,LNV), W1(LMA)
!
      READ(5,*) N, MB
      DO 10 J=1, MB+1
```

```
        READ(5,*) (A(J,I), I=MB-J+2, N)
  10 CONTINUE
!
     WRITE(6,1000) N, MB
     DO 20 J=1, MB+1
        WRITE(FMT,1100) (MB-J+1)*7+1, N-MB+J-1
        WRITE(6,FMT) (A(J,I), I=MB-J+2, N)
  20 CONTINUE
!
     CALL DCSBAA(A,LMA,N,MB,E,VE,LNV,W1,IERR)
!
     WRITE(6,1200) IERR
!
     DO 40 K=1, N-3, 4
        WRITE(6,1300) ('EIGENVALUE ', I=1, 4)
        WRITE(6,1400) (E(I), I=K, K+3)
        WRITE(6,1300) ('EIGENVECTOR', I=1, 4)
        DO 30 J=1, N
           WRITE(6,1500) (VE(J,I), I=K, K+3)
  30    CONTINUE
  40 CONTINUE
     IF(MOD(N,4).NE.0) THEN
        WRITE(6,1300) ('EIGENVALUE ', I=N/4*4+1, N)
        WRITE(6,1400) (E(I), I=N/4*4+1, N)
        WRITE(6,1300) ('EIGENVECTOR', I=N/4*4+1, N)
        DO 50 J=1, N
           WRITE(6,1500) (VE(J,I), I=N/4*4+1, N)
  50    CONTINUE
     ENDIF
     STOP
!
1000 FORMAT(' ',/,/,&
            ' ***  DCSBAA  ***',/,/,&
            '  **  INPUT  **',/,/,&
            '      N = ', I2,/,/,&
            '      BAND WIDTH = ', I2,/,/,&
            '      INPUT MATRIX A',/)
1100 FORMAT('( ,', I3,'X,', I2,'(F7.1))')
1200 FORMAT(' ',/,/,&
            '  **  OUTPUT  **',/,/,&
            '      IERR = ', I4)
1300 FORMAT(' ',/,1X, 4(5X, A11, 2X))
1400 FORMAT(3X, 4(2X, 1PD14.7, 2X))
1500 FORMAT(2X, 4(F14.8, 4X))
     END
```

(d) Output results

```
***  DCSBAA  ***

 **  INPUT  **

     N =  7

     BAND WIDTH =  2

     INPUT MATRIX A

                  1.0    1.0    1.0    1.0    1.0
          -4.0   -4.0   -4.0   -4.0   -4.0   -4.0
    5.0    6.0    6.0    6.0    6.0    6.0    5.0


 **  OUTPUT  **

     IERR =    0

     EIGENVALUE        EIGENVALUE        EIGENVALUE        EIGENVALUE
     2.3177302D-02     3.4314575D-01     1.5243190D+00     4.0000000D+00

     EIGENVECTOR       EIGENVECTOR       EIGENVECTOR       EIGENVECTOR
     0.19134172       -0.35355339       -0.46193977       -0.50000000
     0.35355339       -0.50000000       -0.35355339       -0.00000000
     0.46193977       -0.35355339        0.19134172        0.50000000
     0.50000000       -0.00000000        0.50000000        0.00000000
     0.46193977        0.35355339        0.19134172       -0.50000000
     0.35355339        0.50000000       -0.35355339        0.00000000
     0.19134172        0.35355339       -0.46193977        0.50000000

     EIGENVALUE        EIGENVALUE        EIGENVALUE
     7.6472539D+00     1.1656854D+01     1.4805250D+01

     EIGENVECTOR       EIGENVECTOR       EIGENVECTOR
    -0.46193977       -0.35355339        0.19134172
     0.35355339        0.50000000       -0.35355339
     0.19134172       -0.35355339        0.46193977
    -0.50000000        0.00000000       -0.50000000
     0.19134172        0.35355339        0.46193977
     0.35355339       -0.50000000       -0.35355339
    -0.46193977        0.35355339        0.19134172
```

4.9.2 DCSBAN, RCSBAN
All Eigenvalues of a Real Symmetric Band Matrix

(1) **Function**

DCSBAN or RCSBAN uses the Givens method and root-free QR method to obtain all eigenvalues of the real symmetric band matrix A (symmetric band type).

(2) **Usage**

Double precision:
  CALL DCSBAN (A, LMA, N, MB, E, W1, IERR)
Single precision:
  CALL RCSBAN (A, LMA, N, MB, E, W1, IERR)

(3) **Arguments**

D:Double precision real   Z:Double precision complex   I: $\left\{\begin{matrix}\text{INTEGER(4) as for 32bit Integer}\\\text{INTEGER(8) as for 64bit Integer}\end{matrix}\right\}$
R:Single precision real   C:Single precision complex

No.	Argument	Type	Size	Input/ Output	Contents
1	A	$\left\{\begin{matrix}\text{D}\\\text{R}\end{matrix}\right\}$	LMA, N	Input	Real symmetric band matrix A (symmetric band type) (See Note (a)).
				Output	Input-time contents are not retained.
2	LMA	I	1	Input	Adjustable dimension of array A.
3	N	I	1	Input	Order of matrix A.
4	MB	I	1	Input	Band width of matrix A.
5	E	$\left\{\begin{matrix}\text{D}\\\text{R}\end{matrix}\right\}$	N	Output	Eigenvalues.
6	W1	$\left\{\begin{matrix}\text{D}\\\text{R}\end{matrix}\right\}$	N	Work	Work area
7	IERR	I	1	Output	Error indicator

(4) **Restrictions**

(a) $0 \leq \text{MB} < \text{N}$

(b) $\text{MB} < \text{LMA}$

(5) **Error indicator**

IERR value	Meaning	Processing
0	Normal termination.	
1000	N was equal to 1.	$\mathrm{E}(1) \leftarrow \mathrm{A}(1,1)$ is performed.
1100	MB was equal to 0.	$\mathrm{E}(\mathrm{i}) \leftarrow \mathrm{A}(1,\mathrm{i}) \quad (1 \leq i \leq \mathrm{N})$ is performed.
3000	Restriction (a) or (b) was not satisfied.	Processing is aborted.
$5000 + i$	The sequence did not converge in the step where the eigenvalue is obtained. $(1 \leq i \leq \mathrm{N})$	Eigenvalues obtained by this time are entered in $\mathrm{E}(1), \cdots, \mathrm{E}(\mathrm{i}-1)$ (However, the order is irregular).

(6) **Notes**

(a) The Real symmetric band matrix is compressed into symmetric band type having (MB+1) rows and N columns and stored in array A (See Appendix B).

(b) Eigenvalues are stored in ascending order.

4.9.3 DCSBSS, RCSBSS Eigenvalues and Eigenvectors of a Real Symmetric Band Matrix

(1) **Function**

DCSBSS or RCSBSS uses the Givens method and root-free QR method to obtain the m largest or m smallest eigenvalues of the real symmetric band matrix A (symmetric band type) and the inverse iteration method to obtain the corresponding eigenvectors.

(2) **Usage**

Double precision:

CALL DCSBSS (A, LMA, N, MB, EPS, E, M, VE, LNV, ISW, IW1, W1, IERR)

Single precision:

CALL RCSBSS (A, LMA, N, MB, EPS, E, M, VE, LNV, ISW, IW1, W1, IERR)

(3) **Arguments**

D:Double precision real Z:Double precision complex I: $\left\{ \begin{array}{l} \text{INTEGER(4) as for 32bit Integer} \\ \text{INTEGER(8) as for 64bit Integer} \end{array} \right\}$
R:Single precision real C:Single precision complex

No.	Argument	Type	Size	Input/ Output	Contents
1	A	$\left\{\begin{array}{c}\text{D}\\\text{R}\end{array}\right\}$	LMA, N	Input	Real symmetric band matrix A (symmetric band type) (See Note (a)).
				Output	Input-time contents are not retained.
2	LMA	I	1	Input	Adjustable dimension of array A.
3	N	I	1	Input	Order of matrix A.
4	MB	I	1	Input	Band width of matrix A.
5	EPS	$\left\{\begin{array}{c}\text{D}\\\text{R}\end{array}\right\}$	1	Input	Parameter that assigns an upper limit to the absolute error for use in the eigenvalue convergence test (See Note (d)).
6	E	$\left\{\begin{array}{c}\text{D}\\\text{R}\end{array}\right\}$	M	Output	Eigenvalues.
7	M	I	1	Input	The number m of eigenvalues to be obtained.
8	VE	$\left\{\begin{array}{c}\text{D}\\\text{R}\end{array}\right\}$	LNV, M	Output	Eigenvectors (column vector) corresponding to each eigenvalue.
9	LNV	I	1	Input	Adjustable dimension of array VE.
10	ISW	I	1	Input	Processing switch. ISW $\geq$ 0: Obtain M eigenvalues from the largest one. ISW $<$ 0: Obtain M eigenvalues from the smallest one.
11	IW1	I	M	Output	Eigenvector flag (See Note (e)).

No.	Argument	Type	Size	Input/ Output	Contents
12	W1	$\begin{Bmatrix} \text{D} \\ \text{R} \end{Bmatrix}$	See Contents	Work	Work area **Size**: N × 3 × MB + 6
13	IERR	I	1	Output	Error indicator

(4) **Restrictions**

(a) $0 < \text{N} \leq \text{LNV}$

(b) $0 < \text{M} \leq \text{N}$

(c) $0 \leq \text{MB} < \text{N}$

(d) $\text{MB} < \text{LMA}$

(5) **Error indicator**

IERR value	Meaning	Processing
0	Normal termination.	
1000	N was equal to 1.	$\text{E}(1) \leftarrow \text{A}(1,1)$ and $\text{VE}(1,1) \leftarrow 1.0$ are performed.
1100	MB was equal to 0.	$\text{E(i)} \leftarrow \text{A}(1,\text{i}) \quad (1 \leq i \leq \text{N})$ is performed. 1.0 is entered in appropriate components of each column vector of VE, and 0.0 is entered in remaining components.
2000	The maximum number of iterations was exceeded by the inverse iterations for obtaining eigenvectors.	Some eigenvectors are obtained with low precision, and processing continues. (See Note (e).)
3000	Restriction (a), (b), (c) or (d) was not satisfied.	Processing is aborted.

(6) **Notes**

(a) The real symmetric band matrix is compressed into symmetric band type having (MB+1) rows and N columns and stored in array A (See Appendix B).

(b) If ISW≥0, the eigenvalues are stored in descending order. If ISW<0, they are stored in ascending order.

(c) Eigenvalue calculations are appropriately divided up between the root-free QR method and Bisection method internally.

(d) If EPS ≤ 0, the optimum value is automatically set internally. Normally, a negative value should be set so that this value will be set automatically. EPS is used to obtain eigenvalues by using the Bisection method.

(e) If the maximum number of iterations is exceeded when using the inverse iteration method (IERR = 2000 is output), the following processing is performed.
If IW1(i) = 0: The i-th eigenvector calculation is normally terminated.

If IW1(i) $\neq$ 0: The convergence condition is not satisfied for the i-th eigenvector calculation, and the eigenvector precision is low. In this case, the iteration count is set for IW1(i).
If processing is normally terminated (IERR = 0 is output), IW1(i) = 0 is set.

(f) The eigenvectors are an orthonormal set.

(g) If eigenvectors are not required, use 4.9.4 $\left\{\begin{matrix}\text{DCSBSN}\\ \text{RCSBSN}\end{matrix}\right\}$.

(7) **Example**

(a) Problem

Obtain the three smallest eigenvalues of the matrix:

$$A = \begin{bmatrix} 5 & 2 & 1 & & & & & & & \\ 2 & 6 & 3 & 1 & & & 0 & & & \\ 1 & 3 & 6 & 3 & 1 & & & & & \\ & 1 & 3 & 6 & 3 & 1 & & & & \\ & & 1 & 3 & 6 & 3 & 1 & & & \\ & & & 1 & 3 & 6 & 3 & 1 & & \\ & & & & 1 & 3 & 6 & 3 & 1 & \\ & & 0 & & & 1 & 3 & 6 & 3 & 1 \\ & & & & & & 1 & 3 & 6 & 2 \\ & & & & & & & 1 & 2 & 5 \end{bmatrix}$$

and their corresponding eigenvectors.

(b) Input data

Matrix A, LNA = 11, N = 10, MB = 2, EPS = −1.0, M = 3, LNV = 10 and ISW = −1.

(c) Main program

```
      PROGRAM BCSBSS
! *** EXAMPLE OF DCSBSS ***
      IMPLICIT REAL(8) (A-H,O-Z)
      CHARACTER*80 FMT
      PARAMETER ( LMA = 11, LNV = 11 )
      DIMENSION A(LMA,LMA), E(LMA), VE(LNV,LNV), IW1(LMA), W1(36*LMA)
!
      READ(5,*) N, MB, M
      DO 10 J=1, MB+1
         READ(5,*) (A(J,I), I=MB-J+2, N)
   10 CONTINUE
!
      WRITE(6,1000) N, MB, M
      DO 20 J=1, MB+1
         WRITE(FMT,1100) (MB-J+1)*7+1, N-MB+J-1
         WRITE(6,FMT) (A(J,I), I=MB-J+2, N)
   20 CONTINUE
!
      ISW = -1
      EPS = -1.0D0
!
      CALL DCSBSS(A,LMA,N,MB,EPS,E,M,VE,LNV,ISW,IW1,W1,IERR)
!
      WRITE(6,1200)  IERR
!
      DO 40 K=1, M-3, 4
         WRITE(6,1300) ('EIGENVALUE ', I=1, 4)
         WRITE(6,1400) (E(I), I=K, K+3)
         WRITE(6,1300) ('EIGENVECTOR', I=1, 4)
         DO 30 J=1, N
            WRITE(6,1500) (VE(J,I), I=K, K+3)
   30    CONTINUE
   40 CONTINUE
      IF(MOD(M,4).NE.0) THEN
         WRITE(6,1300) ('EIGENVALUE ', I=M/4*4+1, M)
         WRITE(6,1400) (E(I), I=M/4*4+1, M)
         WRITE(6,1300) ('EIGENVECTOR', I=M/4*4+1, M)
         DO 50 J=1, N
            WRITE(6,1500) (VE(J,I), I=M/4*4+1, M)
   50    CONTINUE
      ENDIF
      STOP
!
 1000 FORMAT(' ',/,/,&
             ' ***  DCSBSS  ***',/,/,&
             '  **  INPUT  **',/,/,&
             '      N = ', I2,/,/,&
             '      BAND WIDTH = ', I2,/,/,&
             '      M = ', I2,/,/,&
             '      INPUT MATRIX A',/)
 1100 FORMAT('( ,', I3,'X,', I2,'(F7.1))')
 1200 FORMAT(' ',/,/,&
             '  **  OUTPUT  **',/,/,&
             '      IERR = ', I4)
 1300 FORMAT(' ',/,1X, 4(5X, A11, 2X))
 1400 FORMAT(3X, 4(2X, 1PD14.7, 2X))
 1500 FORMAT(2X, 4(F14.8, 4X))
      END
```

(d) Output results

```
***  DCSBSS  ***

 **  INPUT  **

     N = 10

     BAND WIDTH =  2

     M =  3

     INPUT MATRIX A

                    1.0    1.0    1.0    1.0    1.0    1.0    1.0    1.0
             2.0    3.0    3.0    3.0    3.0    3.0    3.0    3.0    2.0
      5.0    6.0    6.0    6.0    6.0    6.0    6.0    6.0    6.0    5.0


 **  OUTPUT  **

     IERR =    0

     EIGENVALUE        EIGENVALUE        EIGENVALUE
     1.8799058D+00     1.8926451D+00     2.2578112D+00

     EIGENVECTOR       EIGENVECTOR       EIGENVECTOR
    -0.01172823        0.05600768        0.12429215
    -0.20382326       -0.01048668       -0.41411041
     0.44423970       -0.15306238        0.48738830
    -0.46949161        0.39024431       -0.16106987
     0.20136345       -0.56659903       -0.22265032
     0.20136345        0.56659903        0.22265032
    -0.46949161       -0.39024431        0.16106987
     0.44423970        0.15306238       -0.48738830
    -0.20382326        0.01048668        0.41411041
    -0.01172823       -0.05600768       -0.12429215
```

4.9.4 DCSBSN, RCSBSN Eigenvalues of a Real Symmetric Band Matrix

(1) **Function**

DCSBSN or RCSBSN uses the Givens method and root-free QR method to obtain the m largest or m smallest eigenvalues of the real symmetric band matrix A (symmetric band type).

(2) **Usage**

Double precision:
CALL DCSBSN (A, LMA, N, MB, EPS, E, M, ISW, W1, IERR)
Single precision:
CALL RCSBSN (A, LMA, N, MB, EPS, E, M, ISW, W1, IERR)

(3) **Arguments**

D:Double precision real Z:Double precision complex
R:Single precision real C:Single precision complex
I: $\left\{\begin{array}{l}\text{INTEGER(4) as for 32bit Integer}\\ \text{INTEGER(8) as for 64bit Integer}\end{array}\right\}$

No.	Argument	Type	Size	Input/ Output	Contents
1	A	$\left\{\begin{array}{c}\text{D}\\ \text{R}\end{array}\right\}$	LMA, N	Input	Real symmetric band matrix A (symmetric band type) (See Note (a)).
				Output	Input-time contents are not retained.
2	LMA	I	1	Input	Adjustable dimension of array A.
3	N	I	1	Input	Order of matrix A.
4	MB	I	1	Input	Band width of matrix A.
5	EPS	$\left\{\begin{array}{c}\text{D}\\ \text{R}\end{array}\right\}$	1	Input	Parameter that assigns an upper limit to the absolute error for use in the eigenvalue convergence test (See Note (d)).
6	E	$\left\{\begin{array}{c}\text{D}\\ \text{R}\end{array}\right\}$	M	Output	Eigenvalues.
7	M	I	1	Input	The number of m of eigenvalues to be obtained.
8	ISW	I	1	Input	Processing switch. ISW $\geq$ 0: Obtain M eigenvalues from the largest one. ISW $<$ 0: Obtain M eigenvalues from the smallest one.
9	W1	$\left\{\begin{array}{c}\text{D}\\ \text{R}\end{array}\right\}$	5 × N	Work	Work area
10	IERR	I	1	Output	Error indicator

(4) **Restrictions**

(a) $0 < \mathrm{M} \leq \mathrm{N}$

(b) $0 \leq \mathrm{MB} < \mathrm{N}$

(c) $\mathrm{MB} < \mathrm{LMA}$

(5) **Error indicator**

IERR value	Meaning	Processing
0	Normal termination.	
1000	N was equal to 1.	$\mathrm{E(1)} \leftarrow \mathrm{A}(1,1)$ is performed.
1100	MB was equal to 0.	$\mathrm{E(i)} \leftarrow \mathrm{A}(1,\mathrm{i}) \quad (1 \leq i \leq \mathrm{N})$ is performed.
3000	Restriction (a), (b) or (c) was not satisfied.	Processing is aborted.

(6) **Notes**

(a) The real symmetric band matrix is compressed into symmetric band type having (MB+1) rows and N columns and stored in in array A (See Appendix B).

(b) If ISW ≥ 0, the eigenvalues are stored in descending order. If ISW < 0, they are stored in ascending order.

(c) Eigenvalue calculations are appropriately divided up between the root-free QR method and Bisection method internally.

(d) If EPS ≤ 0, the optimum value is automatically set internally. Normally, a negative value should be set so that this value will be set automatically. EPS is used to obtain eigenvalues by using the Bisection method.

4.9.5 DCSBFF, RCSBFF Eigenvalues and Eigenvectors of a Real Symmetric Band Matrix

(1) **Function**

DCSBFF and RCSBFF uses the subspace method to obtain the eigenvalues having the m largest or m smallest absolute values of the real symmetric band matrix A (symmetric band type) and to obtain the corresponding eigenvectors.

(2) **Usage**

Double precision:

CALL DCSBFF (A, LMA, N, MB, M, ITOL, NITE, E, VE, LNV, MST, IS1, IS2, W1, IW1, IERR)

Single precision:

CALL RCSBFF (A, LMA, N, MB, M, ITOL, NITE, E, VE, LNV, MST, IS1, IS2, W1, IW1, IERR)

(3) **Arguments**

D:Double precision real Z:Double precision complex
R:Single precision real C:Single precision complex
I: $\left\{\begin{matrix}\text{INTEGER(4) as for 32bit Integer}\\\text{INTEGER(8) as for 64bit Integer}\end{matrix}\right\}$

No.	Argument	Type	Size	Input/ Output	Contents
1	A	D R	LMA, N	Input	Real symmetric band matrix A (symmetric band type) (See Appendix B).
				Output	Input-time contents are not retained.
2	LMA	I	1	Input	Adjustable dimension of array A.
3	N	I	1	Input	Order of matrix A.
4	MB	I	1	Input	Band width of matrix A.
5	M	I	1	Input	The number of m of eigenvalues to be obtained.
6	ITOL	I	1	Input	Tolerance used for convergence test (See Note (b)).
7	NITE	I	1	Input	Maximum iteration count (See Note (d)).
8	E	D R	See Contents	Output	Eigenvalues. **Size**: $\min(2 \times \mathrm{M}, \mathrm{N}, \mathrm{M}+8)$
9	VE	D R	See Contents	Output	Eigenvectors (column vector) corresponding to each eigenvalue. **Size**: $(\mathrm{LNV}, \min(2 \times \mathrm{M}, \mathrm{N}, \mathrm{M}+8)$
10	LNV	I	1	Input	Adjustable dimension of array VE.

No.	Argument	Type	Size	Input/ Output	Contents
11	MST	I	1	Output	Number of eigenvalues not calculated (See Note (e)).
12	IS1	I	1	Input	Processing switch. IS1 $\leq$ 0: Obtain eigenvalues having the smallest absolute values. IS1 $>$ 0: Obtain eigenvalues having the largest absolute values.
13	IS2	I	1	Input	Sturm sequence check switch. IS2 $\leq$ 0: Do not check. IS2 $>$ 0: Check.
14	W1	$\begin{Bmatrix} \mathrm{D} \\ \mathrm{R} \end{Bmatrix}$	See Contents	Work	Work area **Size**: $\mathrm{N} \times q + q \times q + 2 \times q + \mathrm{N}$ $q = \min(2 \times \mathrm{M}, \mathrm{N}, \mathrm{M} + 8)$ If IS2 $>$ 0, then N $\times$ (MB $+$ 1) work areas are required.
15	IW1	I	N	Work	Work area
16	IERR	I	1	Output	Error indicator

(4) **Restrictions**

(a) $0 < \mathrm{N} \leq \mathrm{LNV}$

(b) $0 \leq \mathrm{MB} < \mathrm{N}$

(c) $\mathrm{MB} < \mathrm{LMA}$

(d) $0 < \mathrm{M} \leq \mathrm{N}$

(5) **Error indicator**

IERR value	Meaning	Processing
0	Normal termination.	
1000	N was equal to 1.	$\mathrm{E}(1) \leftarrow \mathrm{A}(1,1)$ and $\mathrm{VE}(1,1) \leftarrow 1.0$ are performed.
3000	Restriction (a), (b), (c) or (d) was not satisfied.	Processing is aborted.
4000	An error occurred during processing.	
$5000 + i$	The sequence did not converge within the specified number of iterations.	Processing is aborted after obtaining up to the i-th eigenvalue and eigenvector.

(6) **Notes**

(a) This subroutine is effective when the number of eigenvalues to be obtained is very small (m≪N) relative to the order of the matrix. Otherwise, you should use other subroutines such 4.9.1 $\begin{Bmatrix} \mathrm{DCSBAA} \\ \mathrm{RCSBAA} \end{Bmatrix}$ and

4.9.3 $\left\{\begin{matrix}\text{DCSBSS}\\ \text{RCSBSS}\end{matrix}\right\}$.

(b) This subroutine considers that the eigenvalue has converged if the following condition is satisfied. At this time, the eigenvector has a precision greater than or equal to ITOL/2.
$|\dfrac{a_i^n - a_i^{n-1}}{a_i^n}| \leq 10.0^{-\text{ITOL}}$ (a_i^n: i-th eigenvalue after the n-th iteration)
If the input value of ITOL is less than or equal to 0 or greater than $-$ LOG10 (ε), then the optimum value is automatically set internally. (ε: Unit for determining error)

(c) If ISW ≥ 0, the eigenvalues are stored in descending order. If ISW < 0, they are stored in ascending order.

(d) If the input value of NITE is less than or equal to 0, then 20 is used as the default value.

(e) This subroutine has a function that checks whether the Sturm sequence property was used for the calculated eigenvalues. Although the number of calculated eigenvalues is computed, the number of calculations increases at this time on the order of $\mathrm{N} \times \mathrm{MB}^2$. For example, assume that three eigenvalues having the smallest absolute values are to be obtained for the eigenvalue problem having 6, 5, 3, 2 and 1 as eigenvalues. If 5, 2 and 1 are obtained as solution eigenvalues at this time, then 1 is returned to MST since the value 3 was not obtained as a solution. This function is effective only if all eigenvalues are positive.

(7) **Example**

(a) Problem
Obtain the two eigenvalues having the smallest absolute values of the matrix:

$$A = \begin{bmatrix} 611 & 196 & -192 & 407 & -8 & 0 & 0 & 0 \\ 196 & 899 & 113 & -192 & -71 & -43 & 0 & 0 \\ -192 & 113 & 899 & 196 & 61 & 49 & 8 & 0 \\ 407 & -192 & 196 & 611 & 8 & 44 & 59 & -23 \\ -8 & -72 & 61 & 8 & 411 & -599 & 208 & 208 \\ 0 & -43 & 49 & 44 & -599 & 411 & 208 & 208 \\ 0 & 0 & 8 & 59 & 208 & 208 & 99 & -911 \\ 0 & 0 & 0 & -23 & 208 & 208 & -911 & 99 \end{bmatrix}$$

and their corresponding eigenvectors.

(b) Input data
Matrix A, LMA=11, N=8, MB=4, M=2 and LNV=10.

(c) Main program

```
      PROGRAM BCSBFF
! *** EXAMPLE OF DCSBFF ***
      IMPLICIT REAL(8)(A-H,O-Z)
      CHARACTER*80 FMT
      PARAMETER ( LMA=11, LNV=10, LN=10, LNQ=10 )
      PARAMETER ( LW=LNQ*(LN+LNQ+2)+LN*(LN+1) )
      DIMENSION A(LMA,LN), E(LN), VE(LNV,LNQ), W1(LW), IW1(LN)
!
      READ(5,*) N, MB, M
      DO 10 J=1, MB+1
         READ(5,*) (A(J,I), I=MB-J+2, N)
   10 CONTINUE
!
      WRITE(6,1000) N, MB, M
      DO 20 J=1, MB+1
         WRITE(FMT,1100) (MB-J+1)*8+2, N-MB+J-1
         WRITE(6,FMT) (A(J,I), I=MB-J+2, N)
   20 CONTINUE
!
      CALL DCSBFF(A,LMA,N,MB,M,0,0,E,VE,LNV,MST,0,1,W1,IW1,IERR)
!
```

```
      WRITE(6,1200) IERR
!
      DO 40 K=1, M-3, 4
         WRITE(6,1300) ('EIGENVALUE ', I=1, 4)
         WRITE(6,1400) (E(I), I=K, K+3)
         WRITE(6,1300) ('EIGENVECTOR', I=1, 4)
         DO 30 J=1, N
            WRITE(6,1500) (VE(J,I), I=K, K+3)
   30    CONTINUE
   40 CONTINUE
      IF(MOD(M,4).NE.0) THEN
         WRITE(6,1300) ('EIGENVALUE ', I=M/4*4+1, M)
         WRITE(6,1400) (E(I), I=M/4*4+1, M)
         WRITE(6,1300) ('EIGENVECTOR', I=M/4*4+1, M)
         DO 50 J=1, N
            WRITE(6,1500) (VE(J,I), I=M/4*4+1, M)
   50    CONTINUE
      ENDIF
      WRITE(6,1600) MST
      STOP
!
 1000 FORMAT(' ',/,/,&
            ' ***  DCSBFF  ***',/,/,&
            '  **  INPUT  **',/,/,&
            '      N = ', I2,/,/,&
            '      BAND WIDTH = ', I2,/,/,&
            '      M = ', I2,/,/,&
            '      INPUT MATRIX A',/)
 1100 FORMAT('( ,', I3,'X,', I2,'(F8.1))')
 1200 FORMAT(' ',/,/,&
            '  **  OUTPUT  **',/,/,&
            '      IERR = ', I4)
 1300 FORMAT(' ',/,1X, 4(5X, A11, 2X))
 1400 FORMAT(3X, 4(2X, 1PD14.7, 2X))
 1500 FORMAT(2X, 4(F14.8, 4X))
 1600 FORMAT(' ',/,1X,'     MISSED EIGENVALUES = ', I2)
      END
```

(d) Output results

```
***  DCSBFF  ***

 **  INPUT  **

     N =  8

     BAND WIDTH =  4

     M =  2

     INPUT MATRIX A

                                     -8.0   -43.0     8.0   -23.0
                            407.0   -71.0    49.0    59.0   208.0
                  -192.0   -192.0    61.0    44.0   208.0   208.0
          196.0    113.0   196.0     8.0  -599.0   208.0  -911.0
    611.0   899.0   899.0   611.0   411.0   411.0    99.0    99.0


 **  OUTPUT  **

     IERR =    0

     EIGENVALUE         EIGENVALUE
    -5.9306717D+00      2.2278823D+01

     EIGENVECTOR        EIGENVECTOR
     0.42488478         0.46703726
    -0.26681179        -0.17958149
     0.26654024         0.17944823
    -0.39884494        -0.49603483
    -0.45953441         0.42706413
    -0.43824501         0.44863165
    -0.22420209         0.22834768
    -0.25429564         0.18862045

     MISSED EIGENVALUES =  1
```

4.10 REAL SYMMETRIC TRIDIAGONAL MATRIX (VECTOR TYPE)

4.10.1 DCSTAA, RCSTAA
All Eigenvalues and All Eigenvectors Real Symmetric Tridiagonal Matrix

(1) **Function**

DCSTAA and RCSTAA uses the QR method to obtain all eigenvalues of the real symmetric tridiagonal matrix A (vector type) and all corresponding eigenvectors.

(2) **Usage**

Double precision:
  CALL DCSTAA (D, N, SD, E, VE, LNV, IERR)
Single precision:
  CALL RCSTAA (D, N, SD, E, VE, LNV, IERR)

(3) **Arguments**

D:Double precision real Z:Double precision complex I: $\left\{ \begin{array}{l} \text{INTEGER(4) as for 32bit Integer} \\ \text{INTEGER(8) as for 64bit Integer} \end{array} \right\}$
R:Single precision real C:Single precision complex

No.	Argument	Type	Size	Input/ Output	Contents
1	D	$\left\{\begin{array}{c}\text{D}\\\text{R}\end{array}\right\}$	N	Input	Diagonal components of the real symmetric tridiagonal matrix A.
2	N	I	1	Input	Order of matrix A.
3	SD	$\left\{\begin{array}{c}\text{D}\\\text{R}\end{array}\right\}$	N	Input	Subdiagonal components of the real symmetric tridiagonal matrix A.
				Output	Input-time contents are not retained.
4	E	$\left\{\begin{array}{c}\text{D}\\\text{R}\end{array}\right\}$	N	Output	Eigenvalues.
5	VE	$\left\{\begin{array}{c}\text{D}\\\text{R}\end{array}\right\}$	LNV, N	Output	Eigenvectors (column vector) corresponding to each eigenvalue.
6	LNV	I	1	Input	Adjustable dimension of array VE.
7	IERR	I	1	Output	Error indicator

(4) **Restrictions**

(a) $0 < \text{N} \leq \text{LNV}$

(5) **Error indicator**

IERR value	Meaning	Processing
0	Normal termination.	
1000	N was equal to 1.	$E(1) \leftarrow D(1)$ and $VE(1, 1) \leftarrow 1.0$ are performed.
3000	Restriction (a) was not satisfied.	Processing is aborted.
$5000 + i$	The sequence did not converge in the step where the eigenvalue and eigenvector were to be obtained. $(1 \le i \le N)$	Eigenvalues obtained by this time are entered in $E(1), \cdots, E(i-1)$ and eigenvectors corresponding to them are entered in A (However, the order is irregular).

(6) **Notes**

(a) The diagonal components and subdiagonal components of the real symmetric tridiagonal matrix are stored in the one-dimensional arrays D and SD respectively. SD(N) is arbitrary (See Appendix B).

(b) Eigenvalues are stored in ascending order.

(c) Eigenvectors are an orthonormal set.

(d) If eigenvectors are not required, use 4.10.2 $\left\{\begin{matrix}\text{DCSTAN}\\ \text{RCSTAN}\end{matrix}\right\}$.

(7) **Example**

(a) Problem

Obtain all eigenvalues of the matrix:

$$A = \begin{bmatrix} 4 & 1 & 0 & 0 \\ 1 & 3 & 1 & 0 \\ 0 & 1 & 3 & 1 \\ 0 & 0 & 1 & 4 \end{bmatrix}$$

and their corresponding eigenvectors.

(b) Input data

Diagonal components D of matrix A, N=4, subdiagonal components SDof matrix A and LNV=10.

(c) Main program

```
      PROGRAM BCSTAA
! *** EXAMPLE OF DCSTAA ***
      IMPLICIT REAL(8) (A-H,O-Z)
      PARAMETER (LNV = 10)
      DIMENSION D(LNV), SD(LNV), E(LNV), VE(LNV,LNV)
!
      READ(5,*) N
      READ(5,*) (D(I), I=1, N)
      READ(5,*) (SD(I), I=1, N-1)
!
      WRITE(6,1000) N
      WRITE(6,1100) 'DIAGONAL   ', (D(I), I=1, N)
      WRITE(6,1100) 'SUBDIAGONAL', (SD(I), I=1, N-1)
!
      CALL DCSTAA(D,N,SD,E,VE,LNV,IERR)
!
      WRITE(6,1200) IERR
!
      DO 20 K=1, N-3, 4
         WRITE(6,1300) ('EIGENVALUE ', I=1, 4)
         WRITE(6,1400) (E(I), I=K, K+3)
         WRITE(6,1300) ('EIGENVECTOR', I=1, 4)
         DO 10 J=1, N
            WRITE(6,1500) (VE(J,I), I=K, K+3)
```

```
   10    CONTINUE
   20 CONTINUE
      IF(MOD(N,4).NE.0) THEN
         WRITE(6,1300) ('EIGENVALUE ', I=N/4*4+1, N)
         WRITE(6,1400) (E(I), I=N/4*4+1, N)
         WRITE(6,1300) ('EIGENVECTOR', I=N/4*4+1, N)
         DO 30 J=1, N
            WRITE(6,1500) (VE(J,I), I=N/4*4+1, N)
   30    CONTINUE
      ENDIF
      STOP
!
 1000 FORMAT(' ',/,/,&
             ' ***  DCSTAA  ***',/,/,&
             '  **  INPUT  **',/,/,&
             '      N = ', I2,/,/,&
             '      INPUT MATRIX A')
 1100 FORMAT(' ',/,6X, A11,/,&
             5X, 11(F7.1),/)
 1200 FORMAT(' ',/,/,&
             '  **  OUTPUT  **',/,/,&
             '      IERR = ', I4)
 1300 FORMAT(' ',/,1X, 4(5X, A11, 2X))
 1400 FORMAT(3X, 4(2X, 1PD14.7, 2X))
 1500 FORMAT(2X, 4(F14.8, 4X))
      END
```

(d) Output results

```
***  DCSTAA  ***

 **  INPUT  **

     N =  4

     INPUT MATRIX A

     DIAGONAL
        4.0    3.0    3.0    4.0

     SUBDIAGONAL
        1.0    1.0    1.0


 **  OUTPUT  **

     IERR =    0

     EIGENVALUE        EIGENVALUE        EIGENVALUE        EIGENVALUE
     1.5857864D+00     3.0000000D+00     4.4142136D+00     5.0000000D+00

     EIGENVECTOR       EIGENVECTOR       EIGENVECTOR       EIGENVECTOR
    -0.27059805       -0.50000000       -0.65328148       -0.50000000
     0.65328148        0.50000000       -0.27059805       -0.50000000
    -0.65328148        0.50000000        0.27059805       -0.50000000
     0.27059805       -0.50000000        0.65328148       -0.50000000
```

4.10.2 DCSTAN, RCSTAN
All Eigenvalues of a Real Symmetric Tridiagonal Matrix

(1) **Function**

DCSTAN and RCSTAN uses the QR method to obtain all eigenvalues of the real symmetric tridiagonal matrix A (vector type).

(2) **Usage**

Double precision:
  CALL DCSTAN (D, N, SD, E, IERR)
Single precision:
  CALL RCSTAN (D, N, SD, E, IERR)

(3) **Arguments**

D:Double precision real Z:Double precision complex
R:Single precision real C:Single precision complex
I: $\left\{ \begin{array}{l} \text{INTEGER(4) as for 32bit Integer} \\ \text{INTEGER(8) as for 64bit Integer} \end{array} \right\}$

No.	Argument	Type	Size	Input/ Output	Contents
1	D	$\begin{Bmatrix} \text{D} \\ \text{R} \end{Bmatrix}$	N	Input	Diagonal components of the real symmetric tridiagonal matrix A.
2	N	I	1	Input	Order of matrix A.
3	SD	$\begin{Bmatrix} \text{D} \\ \text{R} \end{Bmatrix}$	N	Input	Subdiagonal components of the real symmetric tridiagonal matrix A.
				Output	Input-time contents are not retained.
4	E	$\begin{Bmatrix} \text{D} \\ \text{R} \end{Bmatrix}$	N	Output	Eigenvalues.
5	IERR	I	1	Output	Error indicator

(4) **Restrictions**

(a) N $>$ 0

(5) **Error indicator**

IERR value	Meaning	Processing
0	Normal termination.	
1000	N was equal to 1.	E(1) ← D(1) is performed.
3000	Restriction (a) was not satisfied.	Processing is aborted.
$5000 + i$	The sequence did not converge in the step where the eigenvalue is obtained. ($1 \leq i \leq$ N)	Eigenvalues obtained by this time are entered in $\text{E}(1), \cdots, \text{E}(\text{i}-1)$ (However, the order is irregular).

(6) **Notes**

(a) The diagonal components and subdiagonal components of the real symmetric tridiagonal matrix are stored in the one-dimensional arrays D and SD respectively. SD(N) is arbitrary (See Appendix B).

(b) Eigenvalues are stored in ascending order.

4.10.3 DCSTSS, RCSTSS Eigenvalues and Eigenvectors of a Real Symmetric Tridiagonal Matrix

(1) **Function**

DCSTSS and RCSTSS uses the root-free QR method or Bisection method to obtain the m largest or m smallest eigenvalues of the real symmetric tridiagonal matrix A (vector type) and the inverse iteration method to obtain the corresponding eigenvectors.

(2) **Usage**

Double precision:

CALL DCSTSS (D, N, SD, EPS, E, M, VE, LNV, ISW, IW1, W1, IERR)

Single precision:

CALL RCSTSS (D, N, SD, EPS, E, M, VE, LNV, ISW, IW1, W1, IERR)

(3) **Arguments**

D:Double precision real Z:Double precision complex
R:Single precision real C:Single precision complex

I: $\left\{\begin{matrix}\text{INTEGER(4) as for 32bit Integer}\\ \text{INTEGER(8) as for 64bit Integer}\end{matrix}\right\}$

No.	Argument	Type	Size	Input/ Output	Contents
1	D	$\begin{Bmatrix}\text{D}\\ \text{R}\end{Bmatrix}$	N	Input	Diagonal components of the real symmetric tridiagonal matrix A.
2	N	I	1	Input	Order of matrix A.
3	SD	$\begin{Bmatrix}\text{D}\\ \text{R}\end{Bmatrix}$	N	Input	Subdiagonal components of the real symmetric tridiagonal matrix A.
4	EPS	$\begin{Bmatrix}\text{D}\\ \text{R}\end{Bmatrix}$	1	Input	Parameter that assigns an upper limit to the absolute error for use in the eigenvalue convergence test (See Note (d)).
5	E	$\begin{Bmatrix}\text{D}\\ \text{R}\end{Bmatrix}$	M	Output	Eigenvalues.
6	M	I	1	Input	The number of m of eigenvalues to be obtained.
7	VE	$\begin{Bmatrix}\text{D}\\ \text{R}\end{Bmatrix}$	LNV, M	Output	Eigenvectors (column vector) corresponding to each eigenvalue.
8	LNV	I	1	Input	Adjustable dimension of array VE.
9	ISW	I	1	Input	Processing switch. ISW $\geq$ 0: Obtain M eigenvalues from the largest one. ISW $<$ 0: Obtain M eigenvalues from the smallest one.
10	IW1	I	M	Output	Eigenvector flag (See Note (e)).
11	W1	$\begin{Bmatrix}\text{D}\\ \text{R}\end{Bmatrix}$	6 × N	Work	Work area
12	IERR	I	1	Output	Error indicator

(4) **Restrictions**

(a) $0 < \mathrm{N} \leq \mathrm{LNV}$

(b) $0 < \mathrm{M} \leq \mathrm{N}$

(5) **Error indicator**

IERR value	Meaning	Processing
0	Normal termination.	
1000	N was equal to 1.	$\mathrm{E}(1) \leftarrow \mathrm{D}(1)$ and $\mathrm{VE}(1,1) \leftarrow 1.0$ are performed.
2000	The maximum number of iterations was exceeded by the inverse iterations for obtaining eigenvectors.	Some eigenvectors are obtained with low precision, and processing continues. (See Note (e).)
3000	Restriction (a) or (b) was not satisfied.	Processing is aborted.

(6) **Notes**

(a) The diagonal components and subdiagonal components of the real symmetric tridiagonal matrix are stored in the one-dimensional arrays D and SD respectively. SD(N) is arbitrary (See Appendix B).

(b) If $\mathrm{ISW} \geq 0$, the eigenvalues are stored in descending order. If $\mathrm{ISW} < 0$, they are stored in ascending order.

(c) Eigenvalue calculations are appropriately divided up between the root-free QR method and Bisection method internally.

(d) If $\mathrm{EPS} \leq 0$, the optimum value is automatically set internally. Normally, a negative value should be set so that this value will be set automatically. EPS is used to obtain eigenvalues by Bisection method.

(e) If the maximum number of iterations is exceeded when using the inverse iteration method (IERR = 2000 is output), the following processing is performed.
If $\mathrm{IW1(i)} = 0$: The i-th eigenvector calculation is normally terminated.
If $\mathrm{IW1(i)} \neq 0$: The convergence condition is not satisfied for the i-th eigenvector calculation, and the eigenvector precision is low. In this case, the iteration count is set for IW1(i).
If processing is normally terminated (IERR = 0 is output), IW1(i) = 0 is set.

(f) Eigenvectors are an orthonormal set.

(g) If eigenvectors are not required, use 4.10.4 $\begin{Bmatrix}\mathrm{DCSTSN}\\ \mathrm{RCSTSN}\end{Bmatrix}$.

(7) **Example**

(a) Problem

Obtain the two largest eigenvalues of the matrix:

$$A = \begin{bmatrix} 5 & 3 & & & & & & & & \\ 3 & 2 & 3 & & & & & & & \\ & 3 & 2 & 3 & & & & 0 & & \\ & & 3 & 2 & 3 & & & & & \\ & & & 3 & 2 & 3 & & & & \\ & & & & 3 & 2 & 3 & & & \\ & & & & & 3 & 2 & 3 & & \\ & & 0 & & & & 3 & 2 & 3 & \\ & & & & & & & 3 & 2 & 3 \\ & & & & & & & & 3 & 5 \end{bmatrix}$$

and their corresponding eigenvectors.

(b) Input data

Diagonal component D of matrix A, N=10, subdiagonal components SD of matrix A, EPS=-1.0 M=2, LNV=10 and ISW=1.

(c) Main program

```
      PROGRAM BCSTSS
! *** EXAMPLE OF DCSTSS ***
      IMPLICIT REAL(8) (A-H,O-Z)
      PARAMETER ( LNV = 10 )
      DIMENSION D(LNV), SD(LNV), E(LNV), VE(LNV,LNV),&
                IW1(LNV), W1(6*LNV)
!
      READ(5,*) N, M
      READ(5,*) (D(I), I=1, N)
      READ(5,*) (SD(I), I=1, N-1)
!
      WRITE(6,1000) N, M
      WRITE(6,1100) 'DIAGONAL    ', (D(I), I=1, N)
      WRITE(6,1100) 'SUBDIAGONAL', (SD(I), I=1, N-1)
!
      ISW = 1
      EPS = -1.0D0
!
      CALL DCSTSS(D,N,SD,EPS,E,M,VE,LNV,ISW,IW1,W1,IERR)
!
      WRITE(6,1200) IERR
!
      DO 20 K=1, M-3, 4
         WRITE(6,1300) ('EIGENVALUE ', I=1, 4)
         WRITE(6,1400) (E(I), I=K, K+3)
         WRITE(6,1300) ('EIGENVECTOR', I=1, 4)
         DO 10 J=1, N
            WRITE(6,1500) (VE(J,I), I=K, K+3)
   10    CONTINUE
   20 CONTINUE
      IF(MOD(M,4).NE.0) THEN
         WRITE(6,1300) ('EIGENVALUE ', I=M/4*4+1, M)
         WRITE(6,1400) (E(I), I=M/4*4+1, M)
         WRITE(6,1300) ('EIGENVECTOR', I=M/4*4+1, M)
         DO 30 J=1, N
            WRITE(6,1500) (VE(J,I), I=M/4*4+1, M)
   30    CONTINUE
      ENDIF
      STOP
!
 1000 FORMAT(' ',/,/,&
             ' ***  DCSTSS  ***',/,/,&
             '  **  INPUT  **',/,/,&
             '      N = ', I2,/,/,&
             '      M = ', I2,/,/,&
             '      INPUT MATRIX A')
 1100 FORMAT(' ',/,6X, A11,/,&
             5X, 11(F7.1),/)
 1200 FORMAT(' ',/,/,&
             '  **  OUTPUT  **',/,/,&
             '      IERR = ', I4)
 1300 FORMAT(' ',/,1X, 4(5X, A11, 2X))
 1400 FORMAT(3X, 4(2X, 1PD14.7, 2X))
 1500 FORMAT(2X, 4(F14.8, 4X))
      END
```

(d) Output results

```
***  DCSTSS  ***

 **  INPUT  **

     N = 10

     M =  2

     INPUT MATRIX A

     DIAGONAL
        5.0    2.0    2.0    2.0    2.0    2.0    2.0    2.0    2.0    5.0

     SUBDIAGONAL
        3.0    3.0    3.0    3.0    3.0    3.0    3.0    3.0    3.0

 **  OUTPUT  **

     IERR =    0

     EIGENVALUE        EIGENVALUE
     8.0000000D+00     7.7063391D+00

     EIGENVECTOR       EIGENVECTOR
     0.31622777       -0.44170765
     0.31622777       -0.39847023
     0.31622777       -0.31622777
     0.31622777       -0.20303072
     0.31622777       -0.06995962
     0.31622777        0.06995962
     0.31622777        0.20303072
     0.31622777        0.31622777
     0.31622777        0.39847023
     0.31622777        0.44170765
```

4.10.4 DCSTSN, RCSTSN Eigenvalues of a Real Symmetric Tridiagonal Matrix

(1) **Function**

DCSTSN and RCSTSN uses the root-free QR method or Bisection method to obtain the m largest or m smallest eigenvalues of the real symmetric tridiagonal matrix A (vector type).

(2) **Usage**

Double precision:
CALL DCSTSN (D, N, SD, EPS, E, M, ISW, W1, IERR)
Single precision:
CALL RCSTSN (D, N, SD, EPS, E, M, ISW, W1, IERR)

(3) **Arguments**

D:Double precision real Z:Double precision complex
R:Single precision real C:Single precision complex
I: $\left\{ \begin{array}{l} \text{INTEGER(4) as for 32bit Integer} \\ \text{INTEGER(8) as for 64bit Integer} \end{array} \right\}$

No.	Argument	Type	Size	Input/ Output	Contents
1	D	$\left\{\begin{matrix} D \\ R \end{matrix}\right\}$	N	Input	Diagonal components of the real symmetric tridiagonal matrix A.
2	N	I	1	Input	Order of matrix A.
3	SD	$\left\{\begin{matrix} D \\ R \end{matrix}\right\}$	N	Input	Subdiagonal components of the real symmetric tridiagonal matrix A.
4	EPS	$\left\{\begin{matrix} D \\ R \end{matrix}\right\}$	1	Input	Parameter that assigns an upper limit to the absolute error for use in the eigenvalue convergence test (See Note (d)).
5	E	$\left\{\begin{matrix} D \\ R \end{matrix}\right\}$	M	Output	Eigenvalues.
6	M	I	1	Input	The number of m of eigenvalues to be obtained.
7	ISW	I	1	Input	Processing switch. ISW $\geq$ 0: Obtain M eigenvalues from the largest one. ISW $<$ 0: Obtain M eigenvalues from the smallest one.
8	W1	$\left\{\begin{matrix} D \\ R \end{matrix}\right\}$	3 × N	Work	Work area
9	IERR	I	1	Output	Error indicator

(4) **Restrictions**

(a) $0 < \mathrm{M} \leq \mathrm{N}$

(5) **Error indicator**

IERR value	Meaning	Processing
0	Normal termination.	
1000	N was equal to 1.	E(1) ← D(1) is performed.
3000	Restriction (a) was not satisfied.	Processing is aborted.

(6) **Notes**

(a) The diagonal components and subdiagonal components of the real symmetric tridiagonal matrix are stored in the one-dimensional arrays D and SD respectively. SD(N) is arbitrary (See Appendix B).

(b) If ISW ≥ 0, the eigenvalues are stored in descending order. If ISW < 0, they are stored in ascending order.

(c) Eigenvalue calculations are appropriately divided up between the root-free QR method and Bisection method internally.

(d) If EPS ≤ 0, the optimum value is automatically set internally. Normally, a negative value should be set so that this value will be set automatically. EPS is used to obtain eigenvalues by Bisection method.

4.10.5 DCSTEE, RCSTEE Eigenvalues in an Interval and Their Eigenvectors of a Real Symmetric Tridiagonal Matrix (Interval Specified)

(1) **Function**

DCSTEE and RCSTEE uses the Bisection method to obtain the m largest or m smallest eigenvalues in a specified interval of the real symmetric tridiagonal matrix A (vector type) and the inverse iteration method to obtain the corresponding eigenvectors.

(2) **Usage**

Double precision:

CALL DCSTEE (D, N, SD, EPS, E, M, E1, E2, VE, LNV, IW1, W1, IERR)

Single precision:

CALL RCSTEE (D, N, SD, EPS, E, M, E1, E2, VE, LNV, IW1, W1, IERR)

(3) **Arguments**

D:Double precision real	Z:Double precision complex	I: $\left\{\begin{array}{l}\text{INTEGER(4) as for 32bit Integer}\\ \text{INTEGER(8) as for 64bit Integer}\end{array}\right\}$
R:Single precision real	C:Single precision complex	

No.	Argument	Type	Size	Input/ Output	Contents
1	D	$\left\{\begin{matrix}D\\R\end{matrix}\right\}$	N	Input	Diagonal components of the real symmetric tridiagonal matrix A.
2	N	I	1	Input	Order of matrix A.
3	SD	$\left\{\begin{matrix}D\\R\end{matrix}\right\}$	N	Input	Subdiagonal components of the real symmetric tridiagonal matrix A.
4	EPS	$\left\{\begin{matrix}D\\R\end{matrix}\right\}$	1	Input	Parameter that assigns an upper limit to the absolute error for use in the eigenvalue convergence test (See Note (b)).
5	E	$\left\{\begin{matrix}D\\R\end{matrix}\right\}$	M	Output	Eigenvalues.
6	M	I	1	Input	Maximum number of the eigenvalues to be computed.
				Output	Number of the obtained eigenvalues.
7	E1	$\left\{\begin{matrix}D\\R\end{matrix}\right\}$	1	Input	E1 < E2: Obtain M eigenvalues in the interval [E1, E2] from the smallest one (E2 is upper bound). E1 > E2: Obtain M eigenvalues in the interval [E1, E2] from the largest one (E2 is lower bound) (See Notes (c) and (d)).
8	E2	$\left\{\begin{matrix}D\\R\end{matrix}\right\}$	1	Input	
9	VE	$\left\{\begin{matrix}D\\R\end{matrix}\right\}$	LNV, M	Output	Eigenvectors (column vector) corresponding to each eigenvalue.
10	LNV	I	1	Input	Adjustable dimension of array VE.
11	IW1	I	M	Output	Eigenvector flag (See Note (e)).

No.	Argument	Type	Size	Input/ Output	Contents
12	W1	$\begin{Bmatrix} \mathrm{D} \\ \mathrm{R} \end{Bmatrix}$	6 × N	Work	Work area
13	IERR	I	1	Output	Error indicator

(4) Restrictions

(a) $0 < \mathrm{N} \leq \mathrm{LNV}$

(b) $0 < \mathrm{M} \leq \mathrm{N}$

(5) Error indicator

IERR value	Meaning	Processing
0	Normal termination.	
1000	N was equal to 1.	$\mathrm{E}(1) \leftarrow \mathrm{D}(1)$ and $\mathrm{VE}(1,1) \leftarrow 1.0$ are performed.
1500	The number of eigenvalues between E1 and E2 is less than M.	All the eigenvalues and the corresponding eigenvectors between E1 and E2 are obtained and the number of the found eigenvalue is output to M.
2000	The maximum number of iterations was exceeded by the inverse iterations for obtaining eigenvectors.	Some eigenvectors are obtained with low precision, and processing continues. (See Note (e).)
3000	Restriction (a) or (b) was not satisfied.	Processing is aborted.

(6) Notes

(a) The diagonal components and subdiagonal components of the real symmetric tridiagonal matrix are stored in the one-dimensional arrays D and SD respectively. SD(N) is arbitrary (See Appendix B).

(b) If EPS ≤ 0, the optimum value is automatically set internally. Normally, a negative value should be set so that this value will be set automatically. EPS is used to obtain eigenvalues by using the Bisection method.

(c) If E1 $<$ E2 the obtained eigenvalues and eigenvectors are stored in ascending order. On the other hand, if E1 $>$ E2 the eigenvalues and eigenvectors are stored in descending order.

(d) If E1 = E2, the eigenvalues in the interval [E1 − EPS, E1 + EPS] are obtained. Normally, E1 should be set to be different E2.

(e) If the maximum number of iterations is exceeded when using the inverse iteration method (IERR = 2000 is output), the following processing is performed.
If IW1(i) = 0: The i-th eigenvector calculation is normally terminated.
If IW1(i) $\neq$ 0: The convergence condition is not satisfied for the i-th eigenvector calculation, and the eigenvector precision is low. In this case, the iteration count is set for IW1(i).
If processing is normally terminated (IERR = 0 is output), IW1(i) = 0 is set.

(f) Eigenvectors are an orthonormal set.

(g) If eigenvectors are not required, use 4.10.6 $\left\{\begin{matrix}\text{DCSTEN}\\ \text{RCSTEN}\end{matrix}\right\}$.

(7) **Example**

(a) Problem
Obtain the four eigenvalues in the interval [0, 7.9] from the largest one of the following symmetric tridiagonal matrix A:

$$A = \begin{bmatrix} 5 & 3 & & & & & & & & \\ 3 & 2 & 3 & & & & & & & \\ & 3 & 2 & 3 & & & & 0 & & \\ & & 3 & 2 & 3 & & & & & \\ & & & 3 & 2 & 3 & & & & \\ & & & & 3 & 2 & 3 & & & \\ & & & & & 3 & 2 & 3 & & \\ & & 0 & & & & 3 & 2 & 3 & \\ & & & & & & & 3 & 2 & 3 \\ & & & & & & & & 3 & 5 \end{bmatrix}$$

and their corresponding eigenvectors.

(b) Input data
Diagonal component D of matrix A, N=10, subdiagonal components SD of matrix A, EPS=−1.0 M=4, E1=7.9, E2=0 and LNV=10.

(c) Main program

```
      PROGRAM BCSTEE
! *** EXAMPLE OF DCSTEE ***
      IMPLICIT REAL(8) (A-H,O-Z)
      PARAMETER ( LNV = 10 )
      DIMENSION D(LNV), SD(LNV), E(LNV), VE(LNV,LNV),&
                IW1(LNV), W1(6*LNV)
!
      READ(5,*) N, M, E1, E2
      READ(5,*) (D(I), I=1, N)
      READ(5,*) (SD(I), I=1, N-1)
!
      WRITE(6,1000) N, M, E1, E2
      WRITE(6,1100) 'DIAGONAL    ', (D(I), I=1, N)
      WRITE(6,1100) 'SUBDIAGONAL', (SD(I), I=1, N-1)
!
      EPS = -1.0D0
!
      CALL DCSTEE(D,N,SD,EPS,E,M,E1,E2,VE,LNV,IW1,W1,IERR)
!
      WRITE(6,1200) IERR
!
      DO 20 K=1, M-3, 4
         WRITE(6,1300) ('EIGENVALUE ', I=1, 4)
         WRITE(6,1400) (E(I), I=K, K+3)
         WRITE(6,1300) ('EIGENVECTOR', I=1, 4)
         DO 10 J=1, N
            WRITE(6,1500) (VE(J,I), I=K, K+3)
   10    CONTINUE
   20 CONTINUE
      IF(MOD(M,4).NE.0) THEN
         WRITE(6,1300) ('EIGENVALUE ', I=M/4*4+1, M)
         WRITE(6,1400) (E(I), I=M/4*4+1, M)
         WRITE(6,1300) ('EIGENVECTOR', I=M/4*4+1, M)
         DO 30 J=1, N
            WRITE(6,1500) (VE(J,I), I=M/4*4+1, M)
   30    CONTINUE
      ENDIF
      STOP
!
 1000 FORMAT(1X,/,/,&
             1X,'***  DCSTEE  ***',/,/,&
             1X,' **  INPUT  **',/,/,&
             1X,'N = ', I4,'  M = ', I4,/,/,&
             1X,'E1= ', 1PD14.7,'  E2= ', 1PD14.7,/,/,&
             1X,'     INPUT MATRIX A')
 1100 FORMAT(1X,/,6X, A11,/,&
             1X, 4X, 11(F7.1),/)
 1200 FORMAT(1X,/,/,&
            1X,' **  OUTPUT  **',/,/,&
            1X,'     IERR = ', I4)
 1300 FORMAT(1X,/,1X, 4(5X, A11, 2X))
 1400 FORMAT(1X, 2X, 4(2X, 1PD14.7, 2X))
 1500 FORMAT(1X, 1X, 4(F14.8, 4X))
      END
```

(d) Output results

```
 ***  DCSTEE  ***

  **  INPUT  **

 N =   10  M =    4

 E1=  7.9000000D+00  E2=  0.0000000D+00

      INPUT MATRIX A

      DIAGONAL
         5.0    2.0    2.0    2.0    2.0    2.0    2.0    2.0    2.0    5.0

      SUBDIAGONAL
         3.0    3.0    3.0    3.0    3.0    3.0    3.0    3.0    3.0

  **  OUTPUT  **

      IERR =    0

      EIGENVALUE        EIGENVALUE        EIGENVALUE        EIGENVALUE
      7.7063391D+00     6.8541020D+00     5.5267115D+00     3.8541020D+00

      EIGENVECTOR       EIGENVECTOR       EIGENVECTOR       EIGENVECTOR
     -0.44170765        0.42532540        0.39847023       -0.36180340
     -0.39847023        0.26286556        0.06995962        0.13819660
     -0.31622777        0.00000000       -0.31622777        0.44721360
     -0.20303072       -0.26286556       -0.44170765        0.13819660
     -0.06995962       -0.42532540       -0.20303072       -0.36180340
      0.06995962       -0.42532540        0.20303072       -0.36180340
      0.20303072       -0.26286556        0.44170765        0.13819660
      0.31622777        0.00000000        0.31622777        0.44721360
      0.39847023        0.26286556       -0.06995962        0.13819660
      0.44170765        0.42532540       -0.39847023       -0.36180340
```

4.10.6 DCSTEN, RCSTEN Eigenvalues in an Interval of a Real Symmetric Tridiagonal Matrix (Interval Specified)

(1) **Function**

DCSTEN and RCSTEN uses the Bisection method to obtain the m largest or m smallest eigenvalues in a specified interval of the real symmetric tridiagonal matrix A (vector type).

(2) **Usage**

Double precision:

CALL DCSTEN (D, N, SD, EPS, E, M, E1, E2, W1, IERR)

Single precision:

CALL RCSTEN (D, N, SD, EPS, E, M, E1, E2, W1, IERR)

(3) **Arguments**

D:Double precision real Z:Double precision complex
R:Single precision real C:Single precision complex
I: $\left\{\begin{matrix}\text{INTEGER(4) as for 32bit Integer}\\\text{INTEGER(8) as for 64bit Integer}\end{matrix}\right\}$

No.	Argument	Type	Size	Input/ Output	Contents
1	D	$\begin{Bmatrix}\text{D}\\\text{R}\end{Bmatrix}$	N	Input	Diagonal components of the real symmetric tridiagonal matrix A.
2	N	I	1	Input	Order of matrix A.
3	SD	$\begin{Bmatrix}\text{D}\\\text{R}\end{Bmatrix}$	N	Input	Subdiagonal components of the real symmetric tridiagonal matrix A.
4	EPS	$\begin{Bmatrix}\text{D}\\\text{R}\end{Bmatrix}$	1	Input	Parameter that assigns an upper limit to the absolute error for use in the eigenvalue convergence test (See Note (b)).
5	E	$\begin{Bmatrix}\text{D}\\\text{R}\end{Bmatrix}$	M	Output	Eigenvalues.
6	M	I	1	Input	Maximum number of the eigenvalues to be computed.
				Output	Number of the obtained eigenvalues.
7	E1	$\begin{Bmatrix}\text{D}\\\text{R}\end{Bmatrix}$	1	Input	E1 < E2: Obtain M eigenvalues in the interval [E1, E2] from the smallest one (E2 is upper bound). E1 > E2: Obtain M eigenvalues in the interval [E1, E2] from the largest one (E2 is lower bound) (See Notes (c) and (d)).
8	E2	$\begin{Bmatrix}\text{D}\\\text{R}\end{Bmatrix}$	1	Input	
9	W1	$\begin{Bmatrix}\text{D}\\\text{R}\end{Bmatrix}$	3 × N	Work	Work area
10	IERR	I	1	Output	Error indicator

(4) **Restrictions**

(a) $0 < \text{M} \leq \text{N}$

(5) **Error indicator**

IERR value	Meaning	Processing
0	Normal termination.	
1000	N was equal to 1.	E(1) ← D(1) is performed.
1500	The number of eigenvalues between E1 and E2 is less than M.	All the eigenvalues and the corresponding eigenvectors between E1 and E2 are obtained and the number of the found eigenvalue is output to M.
3000	Restriction (a) was not satisfied.	Processing is aborted.

(6) **Notes**

(a) The diagonal components and subdiagonal components of the real symmetric tridiagonal matrix are stored in the one-dimensional arrays D and SD respectively. SD(N) is arbitrary (See Appendix B).

(b) If EPS ≤ 0, the optimum value is automatically set internally. Normally, a negative value should be set so that this value will be set automatically. EPS is used to obtain eigenvalues by using the Bisection method.

(c) If E1 $<$ E2 the obtained eigenvalues and eigenvectors are stored in ascending order. On the other hand, if E1 $>$ E2 the eigenvalues and eigenvectors are stored in descending order.

(d) If E1 $=$ E2, the eigenvalues in the interval [E1 $-$ EPS, E1 $+$ EPS] are obtained. Normally, E1 should be set to be different E2.

4.11 REAL SYMMETRIC RANDOM SPARSE MATRIX

4.11.1 DCSRSS, RCSRSS Eigenvalues and Eigenvectors of a Real Symmetric Sparse Matrix (Symmetric One-Dimensional Row-Oriented List Type) (Upper Triangular Type)

(1) **Function**

DCSRSS or RCSRSS uses the Jacobi-Davidson (JD) algorithm to obtain the M extreme (largest or smallest) eigenvalues and -vectors of a real symmetric sparse matrix A (real symmetric one-dimensional row- oriented list)(upper triangular type).

(2) **Usage**

Double precision:

CALL DCSRSS (A, NA, JA, IA, N, X, LDA, E, M, TR, IX, IS, ITM, IPREC, NDIA, ITJD, ITQMR, IW, WK, IERR)

Single precision:

CALL RCSRSS (A, NA, JA, IA, N, X, LDA, E, M, TR, IX, IS, ITM, IPREC, NDIA, ITJD, ITQMR, IW, WK, IERR)

(3) **Arguments**

D:Double precision real Z:Double precision complex

R:Single precision real C:Single precision complex

I: $\left\{\begin{matrix}\text{INTEGER(4) as for 32bit Integer}\\\text{INTEGER(8) as for 64bit Integer}\end{matrix}\right\}$

No.	Argument	Type	Size	Input/ Output	Contents
1	A	$\left\{\begin{matrix}\text{D}\\\text{R}\end{matrix}\right\}$	NA	Input	Real symmetric matrix A (real symmetric one-dimensional row-oriented list type) (upper triangular type) (See Note (a)).
2	NA	I	1	Input	Dimension size of array A: number of the diagonal elements and the nonzero elements in the upper triangle of matrix A.
3	JA	I	NA	Input	JA(i): Column number of i-th element of matrix A
4	IA	I	$\text{N}+1$	Input	IA(i): Element number in array A of the diagonal element of matrix A's i-th row ($\text{IA(N}+1) = \text{IA(N)} + 1$).
5	N	I	1	Input	Order of matrix A.
6	X	$\left\{\begin{matrix}\text{D}\\\text{R}\end{matrix}\right\}$	LDA, M	Input	Initial iteration vectors (if $\text{IX} = 1$).
				Output	Column vectors of IX are eigenvectors.
7	LDA	I	1	Input	Adjustable dimension of array X.
8	E	$\left\{\begin{matrix}\text{D}\\\text{R}\end{matrix}\right\}$	M	Output	Eigenvalues.
9	M	I	1	Input	Number of eigenvalues to be obtained M (See Note (b)).

No.	Argument	Type	Size	Input/ Output	Contents
10	TR	$\begin{Bmatrix} \text{D} \\ \text{R} \end{Bmatrix}$	M	Input	Convergence threshold for the quotient of current residual norm and the initial residual norm (See Note (c)).
				Output	TR(i) $(i = 1, \cdots, \text{M})$: Final residual norms divided by the initial residual norms.
11	IX	I	1	Input/ Output	Switch parameter for selection of initial iteration vectors (See Note (d)). IX $= -1$: No specification of initial iteration vectors; initial eigenvalues and eigenvectors are internally derived from the diagonal of the matrix. IX $= 0$: No specification of initial iteration vectors; random vectors are internally generated. IX $= 1$: Initial iteration vectors are user-specified. Else: Default value **0** is used.
12	IS	I	1	Input	Processing switch (See Note (b)). IS ≥ 0: Obtain M eigenvalues from the largest one (in descending order). IS < 0 : Obtain M eigenvalues from the smallest one (in ascending order).
13	ITM	I	1	Input/ Output	Dimension of subspace (See Note (e)).
14	IPREC	I	1	Input/ Output	Preconditioning method. IPREC $= 0$: Diagonal preconditioning IPREC $= 1$: Iterative QMR preconditioning with NDIA preceding diagonal preconditioning steps. Else: IPREC is reset to default value 1.
15	NDIA	I	1	Input/ Output	Number of preceding diagonal preconditioning steps (See Note (f)).
16	ITJD	I	1	Input/ Output	Maximum number of outer JD iterations (default: 1000) (See Note (g)).
17	ITQMR	I	1	Input/ Output	Maximum number of QMR iterations (default: 1000) (See Note (h)).
18	IW	I	$2 \times \text{M}$	Work	Work area

No.	Argument	Type	Size	Input/ Output	Contents
19	WK	$\begin{Bmatrix} \text{D} \\ \text{R} \end{Bmatrix}$	See Contents	Work	Work area **Size**: $\text{N} \times (2 \times \text{ITM} + 3 \times \text{M} + 9) + \text{ITM} \times (3 \times \text{ITM} + 2) + 4 \times \text{M}$
20	IERR	I	1	Output	Error indicator

(4) **Restrictions**

(a) $\text{N} > 0$

(b) $\text{N} \leq \text{NA}$

(c) $\text{IA}(\text{N}+1) - 1 \leq \text{NA}$

(d) $\text{N} \leq \text{LDA}$

(e) $0 < \text{M} \leq \text{N}$

(f) When $\text{IX} = 1$: (All M user-specified initial iteration vectors) $\neq 0$

(g) When $\text{M} < \text{N}$: $\text{M} < \text{ITM}$

(5) **Error indicator**

IERR value	Meaning	Processing
0	Normal termination.	
1000	N was equal to 1.	$\text{E}(1) \leftarrow \text{A}(1)$ and $\text{X}(1,1) \leftarrow 1.0$ 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.	
3040	Restriction (e) was not satisfied.	
3070	Restriction (f) was not satisfied.	
3100	Restriction (g) was not satisfied.	
5000	Error occurred in course of finding eigenvectors in subspace.	
6000	No convergence to the required accuracy in ITJD iterations. Namely, $\|Ax_i - \lambda_i x_i\|_2 / \|Ax_0 - \lambda_0 x_0\|_2$ is larger than the user-specified threshold.	

(6) **Notes**

(a) The diagonal elements and nonzero elements in the upper triangular portion of the real symmetric matrix are stored rowwise in the one-dimensional array A. (Real symmetric one-dimensional row-oriented list type (upper triangular type); See Appendix B).

(b) If ISW≥0, the M eigenvalues are stored in array E in descending order from the largest one. If ISW<0, the M eigenvalues are stored in array E in ascending order from the smallest one.

(c) Convergence test method depends on the input value TR(1) as follows.

When TR(1) > 0: The input value TR is used as convergence threshold. That is, the convergence condition is as follows.
$\|Ax_i - \lambda_i x_i\|_2 / \|Ax_0 - \lambda_0 x_0\|_2 \leq \mathrm{TR}(1)$

When TR(1) ≤ 0: Convergence threshold is set to the default value 10^{-8} (10^{-5} for single precision). Namely the condition is as follows.
$\|Ax_i - \lambda_i x_i\|_2 / \|Ax_0 - \lambda_0 x_0\|_2 \leq 10^{-8} \quad (10^{-5})$

(d) For IX = 1, the initial iteration vectors are user-specified. Good starting vectors are approximations of the eigenvectors looked for. The user-specified vectors are orthonormalized within this subroutine. If this fails, they are replaced by random starting vectors.

(e) The subspace size ITM is crucial for JD's convergence. ITM must be < M if ITM > M. The maximum value for ITM is the full space size N. For determining a few extreme eigenvalues and -vectors, a subspace size ITM ≥ 2×M is recommended.

Note that the higher the subspace size is chosen, the faster JD's convergence becomes. A larger subspace, however, results in higher memory requirements. For large sparse matrices, subspace sizes of 2×M to 4×M are usually sufficient.

If input value ITM is larger than or equal to N, then ITM is set equal to N and processing continues.

(f) The value of argument NDIA is referred only when IPREC = 1. If IPREC = 1 and the input value NDIA satisfies NDIA < 0, then NDIA is modified to 10 and processing continues. If IPREC = 0, then NDIA is modified to 0, but it is not referred.

(g) If the input value of argument ITJD satisfies ITJD ≤ 0, then ITJD is modified to 1000 and processing continues.

(h) If the input value of argument ITQMR satisfies ITQMR ≤ 0, then ITQMR is modified to 1000 and processing continues.

(i) On output, the eigenvectors are an orthonormal set.

(j) The JD iteration stops when all the residual norms divided by the initial residual norms of all M current eigenvalue and -vector approximations become smaller than or equal to the user-specified threshold which is given as the input value TR(1). The value of the criterion depends on the user's needs. The default value of 10^{-8} (10^{-5} for single precision) should lead to sufficient accuracy in most cases.

(7) **Example**

(a) Problem

Obtain the three smallest eigenvalues of the matrix:

$$A = \begin{bmatrix} 5 & 2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 6 & 3 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 3 & 6 & 3 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 3 & 6 & 3 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 3 & 6 & 3 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 3 & 6 & 3 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 3 & 6 & 3 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 3 & 6 & 3 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 3 & 6 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2 & 5 \end{bmatrix}$$

and their corresponding eigenvectors.

(b) Input data

Arrays defining the matrix A: A, JA, IA.

NA=27, N=10, LDA=11, M=3, TR(1)=1.0D−10, IX=0, IS=−1,

ITM=5, IPREC=1, NDIA=1, ITJD=1000 and ITQMR=1000.

(c) Main program

```
      PROGRAM BCSRSS
! *** EXAMPLE OF DCSRSS ***
      IMPLICIT REAL(8) (A-H,O-Z)
      INTEGER LDA, ITMMAX
      PARAMETER ( LDA = 11, ITMMAX = 11 )
      DIMENSION A(LDA*LDA)
      DIMENSION JA(LDA*LDA),IA(LDA+1)
      DIMENSION X(LDA,LDA),E(LDA),TR(LDA)
      DIMENSION IW(2*ITMMAX)
      DIMENSION WK(LDA*(5*ITMMAX+9)+ITMMAX*(3*ITMMAX+6))
      INTEGER ITM, IPREC, NDIA, ITJD, ITQMR
!
      CHARACTER*80 FMT
!
      READ(5,*)  N ,NA ,M
      WRITE(6,1000) N ,NA ,M
      READ(5,*) (A(I),I=1,NA)
      READ(5,*) (JA(I),I=1,NA)
      READ(5,*) (IA(I),I=1,N+1)
      DO 40 I=1,N
        WRITE(FMT,1100) I*5, IA(I+1)-IA(I)
        WRITE(6,FMT)  ( A(J), J=IA(I), IA(I+1)-1 )
   40 CONTINUE
      WRITE(6,1150)
      WRITE(6,1200) (J,JA(J),J=1,NA)
      WRITE(6,1250)
      WRITE(6,1300) (J,IA(J),J=1,N+1)
!
      ITM = 5
      IX=0
      IS=-1
      IPREC = 1
      NDIA = 1
      ITJD = 1000
      ITQMR = 1000
!
      EPS = 1.0D-10
      TR(1) = EPS
!
      CALL DCSRSS(A, NA, JA, IA, N, X, LDA, E, M, TR, IX, IS,&
                  ITM, IPREC, NDIA, ITJD, ITQMR, IW, WK, IERR)
!
      WRITE(6,1400)  IERR
!
      DO 140 K=1, M-3, 4
         WRITE(6,2300) (’EIGENVALUE ’, J=1,4)
         WRITE(6,2400) (E(J), J=K, K+3)
         WRITE(6,*)
         WRITE(6,2300) (’EIGENVECTOR’, J=1,4)
         DO 130 I=1,N
            WRITE(6,2500) (X(I,J), J=K, K+3)
  130    CONTINUE
         WRITE(6,*)
         WRITE(6,2300) (’RESIDUAL’, J=1,4)
         WRITE(6,2400) (TR(J), J=K, K+3)
  140 CONTINUE
!
      IF(MOD(M,4).NE.0) THEN
         MREM=M/4*4+1
         WRITE(6,2300) (’EIGENVALUE ’, J=MREM, M)
         WRITE(6,2400) (E(J), J=M/4*4+1, M)
         WRITE(6,*)
         WRITE(6,2300) (’EIGENVECTOR’, J=MREM, M)
         DO 150 I=1,N
            WRITE(6,2500) (X(I,J), J=MREM, M)
  150    CONTINUE
         WRITE(6,*)
         WRITE(6,2300) (’RESIDUAL’, I=MREM, M)
         WRITE(6,2400) (TR(J), J=MREM, M)
      ENDIF
      STOP
!
 1000 FORMAT(/,1X,’ ***  DCSRSS  ***’,/,/,&
      1X,’   **  INPUT PARAMETER **’,/,&
      1X,’     N = ’,I5,/,&
      1X,’     NA = ’,I4,/,&
      1X,’     M = ’,I5,/,/,&
      1X,’   ** INPUT MATRIX A **’)
 1100 FORMAT(’(1X,’,I3,’X,’,I2,’(1X,F4.1))’)
 1150 FORMAT(/,1X,’   ** INPUT INDEX JA **’)
 1200 FORMAT((2X,5(’ JA(’,I2,’) = ’,I3,  2X,:)))
 1250 FORMAT(/,1X,’   ** INPUT INDEX IA **’)
 1300 FORMAT((2X,5(’ IA(’,I2,’) = ’,I3,  2X,:)))
 1400 FORMAT(/,/,1X,’   ** OUTPUT **’,/,/,5X,’IERR = ’,I8,/)
```

```
2300 FORMAT(/,1X,4(5X, A11, 2X),/)
2400 FORMAT(1X,'  ',4(2X, 1PD14.7, 2X))
2500 FORMAT(1X,'  ',4(F14.8, 4X))
     END
```

(d) Output results

```
***  DCSRSS  ***

  **  INPUT PARAMETER **
    N =    10
    NA =   27
    M =     3

  ** INPUT MATRIX A **
      5.0  2.0  1.0
           6.0  3.0  1.0
                6.0  3.0  1.0
                     6.0  3.0  1.0
                          6.0  3.0  1.0
                               6.0  3.0  1.0
                                    6.0  3.0  1.0
                                         6.0  3.0  1.0
                                              6.0  2.0
                                                   5.0

  ** INPUT INDEX JA **
 JA( 1) =   1   JA( 2) =    2   JA( 3) =    3   JA( 4) =    2   JA( 5) =    3
 JA( 6) =   4   JA( 7) =    3   JA( 8) =    4   JA( 9) =    5   JA(10) =    4
 JA(11) =   5   JA(12) =    6   JA(13) =    5   JA(14) =    6   JA(15) =    7
 JA(16) =   6   JA(17) =    7   JA(18) =    8   JA(19) =    7   JA(20) =    8
 JA(21) =   9   JA(22) =    8   JA(23) =    9   JA(24) =   10   JA(25) =    9
 JA(26) =  10   JA(27) =   10

  ** INPUT INDEX IA **
 IA( 1) =   1   IA( 2) =    4   IA( 3) =    7   IA( 4) =   10   IA( 5) =   13
 IA( 6) =  16   IA( 7) =   19   IA( 8) =   22   IA( 9) =   25   IA(10) =   27
 IA(11) =  28


  ** OUTPUT **

   IERR =        0


    EIGENVALUE        EIGENVALUE        EIGENVALUE
    1.8799058D+00     1.8926451D+00     2.2578112D+00


    EIGENVECTOR       EIGENVECTOR       EIGENVECTOR
     0.01172823        0.05600768       -0.12429215
     0.20382326       -0.01048668        0.41411041
    -0.44423970       -0.15306238       -0.48738830
     0.46949161        0.39024431        0.16106987
    -0.20136345       -0.56659903        0.22265032
    -0.20136345        0.56659903       -0.22265032
     0.46949161       -0.39024431       -0.16106987
    -0.44423970        0.15306238        0.48738830
     0.20382326        0.01048668       -0.41411041
     0.01172823       -0.05600768        0.12429215


       RESIDUAL          RESIDUAL          RESIDUAL
    9.7859321D-12     3.9447236D-16     5.4323145D-13
```

4.11.2 DCSJSS, RCSJSS Eigenvalues and Eigenvectors of a Real Symmetric Sparse Matrix (Jagged Diagonals Storage Type)

(1) **Function**

DCSJSS or RCSJSS uses the Jacobi-Davidson (JD) algorithm to obtain the M extreme (largest or smallest) eigenvalues and -vectors of a real symmetric matrix A (JAGGED DIAGONALS LIST TYPE)(JAD).

(2) **Usage**

Double precision:

```
CALL DCSJSS (MJAD, AJAD, NA, IAJAD, JAJAD, JADORD, N, X, LDA, E, M, TR,
             IX, IS, ITM, IPREC, NDIA, ITJD, ITQMR, IW, WK, IERR)
```

Single precision:

```
CALL RCSJSS (MJAD, AJAD, NA, IAJAD, JAJAD, JADORD, N, X, LDA, E, M, TR,
             IX, IS, ITM, IPREC, NDIA, ITJD, ITQMR, IW, WK, IERR)
```

(3) **Arguments**

D:Double precision real Z:Double precision complex I: $\left\{\begin{array}{l}\text{INTEGER(4) as for 32bit Integer}\\\text{INTEGER(8) as for 64bit Integer}\end{array}\right\}$
R:Single precision real C:Single precision complex

No.	Argument	Type	Size	Input/ Output	Contents
1	MJAD	I	1	Input	Number of jagged diagonals of matrix A (JAD format) (See Note (a)).
2	AJAD	$\left\{\begin{array}{c}\text{D}\\\text{R}\end{array}\right\}$	NA	Input	Nonzero elements of matrix A (JAD format) (See Note (a)).
3	NA	I	1	Input	Dimension size of array AJAD: number of nonzero elements of matrix A.
4	IAJAD	I	$\text{MJAD} + 1$	Input	IAJAD(i): Starting indices of the i-th jagged diagonal of matrix A in arrays AJAD and JAJAD (See Note (a)).
5	JAJAD	I	NA	Input	JAJAD(i): Column number of the i-th nonzero element of matrix A in AJAD (See Note (a)).
6	JADORD	I	N	Input	The mapping of the left vector ($\boldsymbol{y}$ of $\boldsymbol{y} = A\boldsymbol{x}$) from the real symmetric one-dimensional row-oriented list type ordering to JAD ordering.
7	N	I	1	Input	Order of matrix A.
8	X	$\left\{\begin{array}{c}\text{D}\\\text{R}\end{array}\right\}$	LDA, M	Input	Initial iteration vectors (if $\text{IX} = 1$).
				Output	Column vectors of IX are eigenvectors.
9	LDA	I	1	Input	Adjustable dimension of array X.
10	E	$\left\{\begin{array}{c}\text{D}\\\text{R}\end{array}\right\}$	M	Output	Eigenvalues.

No.	Argument	Type	Size	Input/ Output	Contents
11	M	I	1	Input	Number of eigenvalues to be obtained M (See Note (b)).
12	TR	$\left\{\begin{matrix}\mathrm{D}\\\mathrm{R}\end{matrix}\right\}$	M	Input	Convergence threshold for the quotient of current residual norm and the initial residual norm (See Note (c)).
				Output	TR(i) $(i = 1, 2, \cdots, \mathrm{M})$: Final residual norms divided by the initial residual norms.
13	IX	I	1	Input/ Output	Switch parameter for selection of initial iteration vectors (See Note (d)). IX $= -1$: No specification of initial iteration vectors; initial eigenvalues and eigenvectors are internally derived from the diagonal of the matrix. IX $= 0$: No specification of initial iteration vectors; random vectors are internally generated. IX $= 1$: Initial iteration vectors are user-specified. Else: Default value **0** is used.
14	IS	I	1	Input	Processing switch (See Note (b)). IS ≥ 0: Obtain M eigenvalues from the largest one (in descending order). IS < 0: Obtain M eigenvalues from the smallest one (in ascending order).
15	ITM	I	1	Input/ Output	Dimension of subspace (See Note (e)).
16	IPREC	I	1	Input/ Output	Preconditioning method. IPREC $= 0$: Diagonal preconditioning. IPREC $= 1$: Iterative QMR preconditioning with NDIA preceding diagonal preconditioning steps. Else: IPREC is reset to default value 1.
17	NDIA	I	1	Input/ Output	Number of preceding diagonal preconditioning steps (See Note (f)).
18	ITJD	I	1	Input/ Output	Maximum number of outer JD iterations (default: 1000) (See Note (g)).
19	ITQMR	I	1	Input/ Output	Maximum number of QMR iterations (default: 1000) (See Note (h)).
20	IW	I	$2 \times \mathrm{M}$	Work	Work area

No.	Argument	Type	Size	Input/ Output	Contents
21	WK	$\begin{Bmatrix} \text{D} \\ \text{R} \end{Bmatrix}$	See Contents	Work	Work area **Size**: $\mathrm{N} \times (2 \times \mathrm{ITM} + 3 \times \mathrm{M} + 9) + \mathrm{ITM} \times (3 \times \mathrm{ITM} + 2) + 4 \times \mathrm{M}$
22	IERR	I	1	Output	Error indicator

(4) **Restrictions**

(a) $\mathrm{N} > 0$

(b) $\mathrm{MJAD} \leq \mathrm{N}$

(c) $\mathrm{MJAD} > 0$

(d) $\mathrm{N} \leq \mathrm{NA}$

(e) $\mathrm{IA(N+1)} - 1 \leq \mathrm{NA}$

(f) $\mathrm{N} \leq \mathrm{LDA}$

(g) $0 < \mathrm{M} \leq \mathrm{N}$

(h) When $\mathrm{IX} = 1$: (All M user-specified initial iteration vectors) $\neq 0$

(i) When $\mathrm{M} < \mathrm{N}$: $\mathrm{M} < \mathrm{ITM}$

(5) **Error indicator**

IERR value	Meaning	Processing
0	Normal termination.	
1000	N was equal to 1.	$\mathrm{E}(1) \leftarrow \mathrm{A}(1)$ and $\mathrm{X}(1,1) \leftarrow 1.0$ are performed.
3000	Restriction (a) was not satisfied.	Processing is aborted.
3005	Restriction (b) was not satisfied.	
3007	Restriction (c) was not satisfied.	
3010	Restriction (d) was not satisfied.	
3020	Restriction (e) was not satisfied.	
3030	Restriction (f) was not satisfied.	
3040	Restriction (g) was not satisfied.	
3070	Restriction (h) was not satisfied.	
3100	Restriction (i) was not satisfied.	
5000	Error occurred in course of finding eigenvectors in subspace.	
6000	No convergence to the required accuracy in ITJD iterations. Namely, $\|Ax_i - \lambda_i x_i\|_2 / \|Ax_0 - \lambda_0 x_0\|_2$ is larger than the user-specified threshold.	

(6) **Notes**

(a) The nonzero elements of matrix A are stored as jagged diagonals in the one-dimensional array AJAD. (JAD format; see Section 2005).

(b) If ISW$\geq$0, the M eigenvalues are stored in array E in descending order from the largest one. If ISW<0, the M eigenvalues are stored in array E in ascending order from the smallest one.

(c) Convergence test method depends on the input value TR(1) as follows.

When TR(1) $>$ 0: The input value TR is used as convergence threshold. That is, the convergence condition is as follows.
$\|Ax_i - \lambda_i x_i\|_2 / \|Ax_0 - \lambda_0 x_0\|_2 \leq \mathrm{TR10}$

When TR(1) $\leq$ 0: Convergence threshold is set to the default value 10^{-8} (10^{-5} for single precision). Namely the condition is as follows.
$\|Ax_i - \lambda_i x_i\|_2 / \|Ax_0 - \lambda_0 x_0\|_2 \leq 10^{-8}$ (10^{-5})

(d) For IX = 1, the initial iteration vectors are user-specified. Good starting vectors are approximations of the eigenvectors looked for. The user-specified vectors are orthonormalized within this subroutine. If this fails, they are replaced by random starting vectors.

(e) The subspace size ITM is crucial for JD's convergence. ITM must be > M if M < N. The maximum value for ITM is the full space size N. For determining a few extreme eigenvalues and -vectors, a subspace size ITM $\geq$ 2×M is recommended.

Note that the higher the subspace size is chosen, the faster JD's convergence becomes. A larger subspace, however, results in higher memory requirements. For large sparse matrices, subspace sizes of 2×M to 4×M are usually sufficient.

If input value ITM is larger than or equal to N, then ITM is set equal to N and processing continues.

(f) The value of argument NDIA is referred only when IPREC = 1. If IPREC = 1 and the input value NDIA satisfies NDIA < 0, then NDIA is modified to 10 and processing continues. If IPREC = 0, then NDIA is modified to 0, but it is not referred.

(g) If the input value of argument ITJD satisfies ITJD $\leq$ 0, then ITJD is modified to 1000 and processing continues.

(h) If the input value of argument ITQMR satisfies ITQMR $\leq$ 0, then ITQMR is modified to 1000 and processing continues.

(i) On output, the eigenvectors are an orthonormal set.

(j) The JD iteration stops when all the residual norms divided by the initial residual norms of all M current eigenvalue and -vector approximations become smaller than or equal to the user-specified threshold which is given as the input value TR(1). The value of the criterion depends on the user's needs. The default value of 10^{-8} (10^{-5} for single precision) should lead to sufficient accuracy in most cases.

(7) **Example**

(a) Problem

Obtain the three smallest eigenvalues of the matrix:

$$A = \begin{bmatrix} 5 & 2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 6 & 3 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 3 & 6 & 3 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 3 & 6 & 3 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 3 & 6 & 3 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 3 & 6 & 3 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 3 & 6 & 3 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 3 & 6 & 3 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 3 & 6 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2 & 5 \end{bmatrix}$$

and their corresponding eigenvectors.

(b) Input data

Arrays defining the matrix A: A, JA and IA.

Converted to: MJAD, AJAD, IAJAD, JAJAD and JADORD (JAD format).

NA=27, N=10, LDA=11, M=3, TR(1)=1.0D−10, IX=0, IS=−1,

ITM=5, IPREC=1, NDIA=1, ITJD=1000 and ITQMR=1000.

(c) Main program

```
      PROGRAM BCSJSS
! *** EXAMPLE OF DCSJSS ***
      IMPLICIT NONE
      INTEGER N, NA, NAJAD, M, I, J, K, IERR
      INTEGER LDA, ITMMAX, LXA, LXIA
      PARAMETER ( LDA = 11, ITMMAX = 11 )
      PARAMETER ( LXA = LDA*LDA, LXIA = LDA+1 )
      REAL(8) A(LDA*LDA)
      INTEGER JA(LDA*LDA), IA(LDA+1)
!
      REAL(8) AJAD(LXA)
      INTEGER IAJAD(LXIA), JAJAD(LXA), JADORD(LDA), MJAD
      INTEGER IWJ(LDA*3+1)
!
      REAL(8) X(LDA,LDA), E(LDA), TR(LDA), EPS
      INTEGER IW(2*ITMMAX)
      REAL(8) WK(LDA*(5*ITMMAX+11)+ITMMAX*(3*ITMMAX+6))
      INTEGER ITM, IPREC, NDIA, ITJD, ITQMR, IX, IS, MREM
!
      CHARACTER*80 FMT
!
      READ(5,*)  N, NA, M
      WRITE(6,1000) N ,NA ,M
      READ(5,*) (A(I),I=1,NA)
      READ(5,*) (JA(I),I=1,NA)
      READ(5,*) (IA(I),I=1,N+1)
      DO 40 I=1,N
        WRITE(FMT,1100) I*5, IA(I+1)-IA(I)
        WRITE(6,FMT)  ( A(J), J=IA(I), IA(I+1)-1 )
   40 CONTINUE
      WRITE(6,1150)
      WRITE(6,1200) (J,JA(J),J=1,NA)
      WRITE(6,1250)
      WRITE(6,1300) (J,IA(J),J=1,N+1)
!
      CALL DARSJD(N, A, IA, JA, LXA, LXIA,&
           MJAD, AJAD, IAJAD, JAJAD,&
           JADORD, IWJ, IERR)
!
      WRITE(6,1400) IERR
!
      ITM = 5
      IX = 0
      IS = -1
      IPREC = 1
      NDIA = 1
      ITJD = 1000
      ITQMR = 1000
!
```

```
     EPS = 1.0D-10
     TR(1) = EPS

     NAJAD = IAJAD(MJAD+1) - IAJAD(1)
!
     CALL DCSJSS(MJAD, AJAD, NAJAD, IAJAD, JAJAD, JADORD,&
                 N, X, LDA, E, M, TR, IX, IS,&
                 ITM, IPREC, NDIA, ITJD, ITQMR, IW, WK, IERR)
!
     WRITE(6,1500)  IERR
!
     DO 140 K=1, M-3, 4
        WRITE(6,2300) ('EIGENVALUE ', J=1,4)
        WRITE(6,2400) (E(J), J=K, K+3)
        WRITE(6,2300) ('EIGENVECTOR', J=1,4)
        DO 130 I=1,N
           WRITE(6,2500) (X(I,J), J=K, K+3)
 130    CONTINUE
        WRITE(6,2300) ('RESIDUAL   ', J=1,4)
        WRITE(6,2400) (TR(J), J=K, K+3)
 140 CONTINUE
!
     IF(MOD(M,4).NE.0) THEN
        MREM=M/4*4+1
        WRITE(6,2300) ('EIGENVALUE ', J=MREM, M)
        WRITE(6,2400) (E(J), J=M/4*4+1, M)
        WRITE(6,2300) ('EIGENVECTOR', J=MREM, M)
        DO 150 I=1,N
           WRITE(6,2500) (X(I,J), J=MREM, M)
 150    CONTINUE
        WRITE(6,2300) ('RESIDUAL   ', I=MREM, M)
        WRITE(6,2400) (TR(J), J=MREM, M)
     ENDIF
     STOP
!
1000 FORMAT(/,1X,' ***  DCSJSS  ***',/,/,&
     1X,'   ** INPUT PARAMETER **',/,&
     1X,'     N = ',I5,/,&
     1X,'     NA = ',I4,/,&
     1X,'     M = ',I5,/,/,&
     1X,'   ** INPUT MATRIX A **')
1100 FORMAT('(1X,',I3,'X,',I2,'(1X,F4.1))')
1150 FORMAT(/,1X,'   ** INPUT INDEX JA **')
1200 FORMAT((2X,5(' JA(',I2,') = ',I3,  2X,:)))
1250 FORMAT(/,1X,'   ** INPUT INDEX IA **')
1300 FORMAT((2X,5(' IA(',I2,') = ',I3,  2X,:)))
1400 FORMAT(/,/,&
     1X,'    IERR AT DARSJD (STORAGE TRANSFORM CSR->JAD) = ',I6)
1500 FORMAT(/,/,1X,'   ** OUTPUT **',/,/,5X,'IERR = ',I8)
2300 FORMAT(/,1X,4(5X, A11, 2X),/)
2400 FORMAT(1X,'  ',4(2X, 1PD14.7, 2X))
2500 FORMAT(1X,'  ',4(F14.8, 4X))
     END
```

(d) Output results

```
***  DCSJSS  ***

  ** INPUT PARAMETER **
    N =    10
    NA =   27
    M =     3

  ** INPUT MATRIX A **
      5.0  2.0  1.0
           6.0  3.0  1.0
                6.0  3.0  1.0
                     6.0  3.0  1.0
                          6.0  3.0  1.0
                               6.0  3.0  1.0
                                    6.0  3.0  1.0
                                         6.0  3.0  1.0
                                              6.0  2.0
                                                   5.0

  ** INPUT INDEX JA **
 JA( 1) =   1   JA( 2) =   2   JA( 3) =   3   JA( 4) =   2   JA( 5) =   3
 JA( 6) =   4   JA( 7) =   3   JA( 8) =   4   JA( 9) =   5   JA(10) =   4
 JA(11) =   5   JA(12) =   6   JA(13) =   5   JA(14) =   6   JA(15) =   7
 JA(16) =   6   JA(17) =   7   JA(18) =   8   JA(19) =   7   JA(20) =   8
 JA(21) =   9   JA(22) =   8   JA(23) =   9   JA(24) =  10   JA(25) =   9
 JA(26) =  10   JA(27) =  10

  ** INPUT INDEX IA **
 IA( 1) =   1   IA( 2) =   4   IA( 3) =   7   IA( 4) =  10   IA( 5) =  13
 IA( 6) =  16   IA( 7) =  19   IA( 8) =  22   IA( 9) =  25   IA(10) =  27
 IA(11) =  28


   IERR AT DARSJD (STORAGE TRANSFORM CSR->JAD) =      0


  ** OUTPUT **
```

```
IERR =        0

 EIGENVALUE          EIGENVALUE          EIGENVALUE
 1.8799058D+00       1.8926451D+00       2.2578112D+00

 EIGENVECTOR         EIGENVECTOR         EIGENVECTOR
 -0.01172823          0.05600768         -0.12429215
 -0.20382326         -0.01048668          0.41411041
  0.44423970         -0.15306238         -0.48738830
 -0.46949161          0.39024431          0.16106987
  0.20136345         -0.56659903          0.22265032
  0.20136345          0.56659903         -0.22265032
 -0.46949161         -0.39024431         -0.16106987
  0.44423970          0.15306238          0.48738830
 -0.20382326          0.01048668         -0.41411041
 -0.01172823         -0.05600768          0.12429215

 RESIDUAL            RESIDUAL            RESIDUAL
 9.7858582D-12       6.8865735D-16       5.4312008D-13
```

4.12 COMPLEX HERMITIAN RANDOM SPARSE MATRIX

4.12.1 ZCHJSS, CCHJSS

Eigenvalues and Eigenvectors of a Complex Hermitian Sparse Matrix (JAD; Jagged Diagonals Storage Type)

(1) **Function**

ZCHJSS or CCHJSS uses the Jacobi-Davidson (JD) algorithm to obtain the M extreme (largest or smallest) eigenvalues and eigenvectors of a complex Hermitian matrix A (SPARSE JAGGED DIAGONALS LIST TYPE)(JAD).

(2) **Usage**

Double precision:

```
CALL ZCHJSS (MJAD, AJAD, NA, IAJAD, JAJAD, JADORD, N, X, LDA, E, M, TR,
             IX, IS, ITM, IPREC, NDIA, ITJD, ITQMR, IW, WK, IERR)
```

Single precision:

```
CALL CCHJSS (MJAD, AJAD, NA, IAJAD, JAJAD, JADORD, N, X, LDA, E, M, TR,
             IX, IS, ITM, IPREC, NDIA, ITJD, ITQMR, IW, WK, IERR)
```

(3) **Arguments**

D:Double precision real Z:Double precision complex I: $\left\{\begin{matrix}\text{INTEGER(4) as for 32bit Integer}\\\text{INTEGER(8) as for 64bit Integer}\end{matrix}\right\}$
R:Single precision real C:Single precision complex

No.	Argument	Type	Size	Input/ Output	Contents
1	MJAD	I	1	Input	Number of jagged diagonals of matrix A (JAD format) (See Note (a)).
2	AJAD	$\left\{\begin{matrix}\text{Z}\\\text{C}\end{matrix}\right\}$	NA	Input	Nonzero elements of matrix A (JAD format) (See Note (a)).
3	NA	I	1	Input	Dimension size of array AJAD: number of nonzero elements of matrix A.
4	IAJAD	I	$\text{MJAD}+1$	Input	IAJAD(i): Starting indices of the i-th jagged diagonal of matrix A in arrays AJAD and JAJAD (See Note (a)).
5	JAJAD	I	NA	Input	JAJAD(i): Column number of the i-th nonzero element of matrix A in AJAD (See Note (a)).
6	JADORD	I	N	Input	The mapping of the left vector ($\boldsymbol{y}$ of $\boldsymbol{y} = A\boldsymbol{x}$ from the real symmetric one-dimensional row-oriented list type ordering to JAD ordering.
7	N	I	1	Input	Order of matrix A.
8	X	$\left\{\begin{matrix}\text{Z}\\\text{C}\end{matrix}\right\}$	LDA, M	Input	Initial iteration vectors (if $\text{IX} = 1$).
				Output	Column vectors of IX are eigenvectors.

No.	Argument	Type	Size	Input/ Output	Contents
9	LDA	I	1	Input	Adjustable dimension of array X.
10	E	$\left\{\begin{matrix}\mathrm{D}\\\mathrm{R}\end{matrix}\right\}$	M	Output	Eigenvalues.
11	M	I	1	Input	Number of eigenvalues to be obtained M (See Note (b)).
12	TR	$\left\{\begin{matrix}\mathrm{D}\\\mathrm{R}\end{matrix}\right\}$	M	Input	Convergence threshold for the quotient of current residual norm and the initial residual norm (See Note (c)).
				Output	TR(i) $(i = 1, \cdots, \mathrm{M})$: Final residual norms divided by the initial residual norms.
13	IX	I	1	Input/ Output	Switch parameter for selection of initial iteration vectors (See Note (d)). IX $= -1$: No specification of initial iteration vectors; initial eigenvalues and eigenvectors are internally derived from the diagonal of the matrix. IX $= 0$: No specification of initial iteration vectors; random vectors are internally generated. IX $= 1$: Initial iteration vectors are user-specified. Else: Default value **0** is used.
14	IS	I	1	Input	Processing switch (See Note (b)). IS ≥ 0: Obtain M eigenvalues from the largest one (in descending order). IS < 0: Obtain M eigenvalues from the smallest one (in ascending order).
15	ITM	I	1	Input/ Output	Dimension of subspace (See Note (e)).
16	IPREC	I	1	Input/ Output	Preconditioning method. IPREC $= 0$: Diagonal preconditioning. IPREC $= 1$: Iterative QMR preconditioning with NDIA preceding diagonal preconditioning steps. Else: IPREC is reset to default value 1.
17	NDIA	I	1	Input/ Output	Number of preceding diagonal preconditioning steps (See Note (f)).
18	ITJD	I	1	Input/ Output	Maximum number of outer JD iterations (default: 1000) (See Note (g)).
19	ITQMR	I	1	Input/ Output	Maximum number of QMR iterations (default: 1000) (See Note (h)).

No.	Argument	Type	Size	Input/ Output	Contents
20	IW	I	$2 \times \mathrm{M}$	Work	Work area
21	WK	$\begin{Bmatrix} \mathrm{Z} \\ \mathrm{C} \end{Bmatrix}$	See Contents	Work	Work area **Size**: $\mathrm{N} \times (2 \times \mathrm{ITM} + 3 \times \mathrm{M} + 9) + \mathrm{ITM} \times (3 \times \mathrm{ITM} + 2) + 4 \times \mathrm{M}$
22	IERR	I	1	Output	Error indicator

(4) Restrictions

(a) $\mathrm{N} > 0$

(b) $\mathrm{MJAD} \le \mathrm{N}$

(c) $\mathrm{MJAD} > 0$

(d) $\mathrm{N} \le \mathrm{NA}$

(e) $\mathrm{IAJAD(MJAD + 1) - 1} \le \mathrm{NA}$

(f) $\mathrm{N} \le \mathrm{LDA}$

(g) $0 < \mathrm{M} \le \mathrm{N}$

(h) When $\mathrm{IX} = 1$: (All M user-specified initial iteration vectors) $\neq 0$

(i) When $\mathrm{M} < \mathrm{N}$: $\mathrm{M} < \mathrm{ITM}$

(5) Error indicator

IERR value	Meaning	Processing
0	Normal termination.	
1000	N was equal to 1.	$\mathrm{E}(1) \leftarrow \mathrm{A}(1)$ and $\mathrm{X}(1,1) \leftarrow 1.0$ are performed.
3000	Restriction (a) was not satisfied.	Processing is aborted.
3005	Restriction (b) was not satisfied.	
3007	Restriction (c) was not satisfied.	
3010	Restriction (d) was not satisfied.	
3020	Restriction (e) was not satisfied.	
3030	Restriction (f) was not satisfied.	
3040	Restriction (g) was not satisfied.	
3070	Restriction (h) was not satisfied.	
3100	Restriction (i) was not satisfied.	
5000	Error occurred in course of finding eigenvectors in subspace.	
6000	No convergence to the required accuracy in ITJD iterations. Namely, $\|Ax_i - \lambda_i x_i\|_2 / \|Ax_0 - \lambda_0 x_0\|_2$ is larger than the user-specified threshold.	

(6) **Notes**

(a) The nonzero elements of matrix A are stored as jagged diagonals in the one-dimensional array AJAD. (JAD format; see Section 2005).

(b) If ISW≥0, the M eigenvalues are stored in array E in descending order from the largest one. If ISW<0, the M eigenvalues are stored in array E in ascending order from the smallest one.

(c) Convergence test method depends on the input value TR(1) as follows.

When $\mathrm{TR}(1) > 0$: The input value TR is used as convergence threshold. That is, the convergence condition is as follows.
$\|Ax_i - \lambda_i x_i\|_2 / \|Ax_0 - \lambda_0 x_0\|_2 \le \mathrm{TR}(1)$

When $\mathrm{TR}(1) \le 0$: Convergence threshold is set to the default value 10^{-8} (10^{-5} for single precision). Namely the condition is as follows.
$\|Ax_i - \lambda_i x_i\|_2 / \|Ax_0 - \lambda_0 x_0\|_2 \le 10^{-8} \quad (10^{-5})$

(d) For IX = 1, the initial iteration vectors are user-specified. Good starting vectors are approximations of the eigenvectors looked for. The user-specified vectors are orthonormalized within this subroutine. If this fails, they are replaced by random starting vectors.

(e) The subspace size ITM is crucial for JD's convergence. ITM must be > M if M < N. The maximum value for ITM is the full space size N. For determining a few extreme eigenvalues and -vectors, a subspace size ITM ≥ 2×M is recommended.

Note that the higher the subspace size is chosen, the faster JD's convergence becomes. A larger subspace, however, results in higher memory requirements. For large sparse matrices, subspace sizes of 2×M to 4×M are usually sufficient.

If input value ITM is larger than or equal to N, then ITM is set equal to N and processing continues.

(f) The value of argument NDIA is referred only when IPREC = 1. If IPREC = 1 and the input value NDIA satisfies NDIA < 0, then NDIA is modified to 10 and processing continues. If IPREC = 0, then NDIA is modified to 0, but it is not referred.

(g) If the input value of argument ITJD satisfies ITJD ≤ 0, then ITJD is modified to 1000 and processing continues.

(h) If the input value of argument ITQMR satisfies ITQMR ≤ 0, then ITQMR is modified to 1000 and processing continues.

(i) On output, the eigenvectors are an orthonormal set.

(j) The JD iteration stops when all the residual norms divided by the initial residual norms of all M current eigenvalue and -vector approximations become smaller than or equal to the user-specified threshold which is given as the input value TR(1). The value of the criterion depends on the user's needs. The default value of 10^{-8} (10^{-5} for single precision) should lead to sufficient accuracy in most cases.

(7) **Example**

(a) Problem

Obtain the two smallest eigenvalues of the matrix:

$$A = \begin{bmatrix} -2.28 & 1.78 - 2.03i & 2.26 + 0.10i & -0.12 + 2.53i \\ 1.78 + 2.03i & -1.12 & 0.01 + 0.43i & -1.07 + 0.86i \\ 2.26 - 0.10i & 0.01 - 0.43i & -0.37 & 2.31 - 0.92i \\ -0.12 - 2.53i & -1.07 - 0.86i & 2.31 + 0.92i & -0.7300 \end{bmatrix}$$

and their corresponding eigenvectors.

(b) Input data

Arrays defining the matrix A: A, JA and IA.

Converted to: MJAD, AJAD, IAJAD, JAJAD and JADORD (JAD format).

NA=121, N=4, LDA=11, M=2, TR(1)=1.0D−10, IX=0, IS=−1,

ITM=3, IPREC=1, NDIA=1, ITJD=1000 and ITQMR=1000.

(c) Main program

```
      PROGRAM ACHJSS
! *** EXAMPLE OF ZCHJSS ***
      IMPLICIT NONE
      INTEGER N, NA, NAJAD, M, I, J, K, IERR
      INTEGER LDA, ITMMAX, LXA, LXIA
      PARAMETER ( LDA = 11, ITMMAX = 11 )
      PARAMETER ( LXA = LDA*LDA, LXIA = LDA+1 )
      COMPLEX(8) A(LDA*LDA)
      INTEGER JA(LDA*LDA),IA(LDA+1)
!
      COMPLEX(8) AJAD(LXA)
      INTEGER IAJAD(LXIA), JAJAD(LXA), JADORD(LDA), MJAD
      INTEGER IWJ(LDA*3+1)
!
      COMPLEX(8) X(LDA,LDA)
      REAL(8) E(LDA),TR(LDA), EPS
      INTEGER IW(2*ITMMAX)
      COMPLEX(8) WK(LDA*(5*ITMMAX+11)+ITMMAX*(3*ITMMAX+6))
      INTEGER ITM, IPREC, NDIA, ITJD, ITQMR, IX, IS, MREM
!
      CHARACTER*80 FMT
!
      READ(5,*)  N, NA, M
      WRITE(6,1000) N ,NA ,M
      READ(5,*) (A(I),I=1,NA)
      READ(5,*) (JA(I),I=1,NA)
      READ(5,*) (IA(I),I=1,N+1)
      DO 40 I=1,N
        WRITE(FMT,1100) (I-1)*18+1, IA(I+1)-IA(I)
        WRITE(6,FMT)  ( A(J), J=IA(I), IA(I+1)-1 )
   40 CONTINUE
      WRITE(6,1150)
      WRITE(6,1200) (J,JA(J),J=1,NA)
      WRITE(6,1250)
      WRITE(6,1300) (J,IA(J),J=1,N+1)
!
      CALL ZARSJD(N, A, IA, JA, LXA, LXIA,&
           MJAD, AJAD, IAJAD, JAJAD,&
           JADORD, IWJ, IERR)
!
      WRITE(6,1400) IERR
!
      ITM = 5
      IX = 0
      IS = -1
      IPREC = 1
      NDIA = 1
      ITJD = 1000
      ITQMR = 1000
!
      EPS = 1.0D-10
      TR(1) = EPS

      NAJAD = IAJAD(MJAD+1) - IAJAD(1)
!
      CALL ZCHJSS(MJAD, AJAD, NAJAD, IAJAD, JAJAD, JADORD,&
                  N, X, LDA, E, M, TR, IX, IS,&
                  ITM, IPREC, NDIA, ITJD, ITQMR, IW, WK, IERR)
!
      WRITE(6,1500)  IERR
!
      DO 140 K=1, M-1, 2
         WRITE(6,2300) ('EIGENVALUE ', J=1,2)
         WRITE(6,2400) (E(J), J=K, K+1)
         WRITE(6,2300) ('EIGENVECTOR', J=1,2)
         DO 130 I=1,N
            WRITE(6,2500) (X(I,J), J=K, K+1)
  130    CONTINUE
         WRITE(6,2300) ('RESIDUAL   ', J=1,2)
         WRITE(6,2400) (TR(J), J=K, K+1)
  140 CONTINUE
!
      IF(MOD(M,2).NE.0) THEN
         MREM=M/2*2+1
         WRITE(6,2300) ('EIGENVALUE ', J=MREM, M)
         WRITE(6,2400) (E(J), J=M/2*2+1, M)
         WRITE(6,2300) ('EIGENVECTOR', J=MREM, M)
         DO 150 I=1,N
            WRITE(6,2500) (X(I,J), J=MREM, M)
  150    CONTINUE
         WRITE(6,2300) ('RESIDUAL   ', I=MREM, M)
         WRITE(6,2400) (TR(J), J=MREM, M)
      ENDIF
      STOP
!
 1000 FORMAT(/,1X,’ ***  ZCHJSS  ***’,/,/,&
      1X,’   ** INPUT PARAMETER **’,/,&
      1X,’     N = ’,I5,/,&
      1X,’     NA = ’,I4,/,&
      1X,’     M = ’,I5,/,/,&
      1X,’   ** INPUT MATRIX A **’)
 1100 FORMAT(’(1X,’,I3,’X,’,I2,&
       ’(4X,"(",F5.2,",",1X,F5.2,")"))’)
 1150 FORMAT(/,1X,’   ** INPUT INDEX JA **’)
 1200 FORMAT((2X,5(’ JA(’,I2,’) = ’,I3,  2X,:)))
 1250 FORMAT(/,1X,’   ** INPUT INDEX IA **’)
 1300 FORMAT((2X,5(’ IA(’,I2,’) = ’,I3,  2X,:)))
 1400 FORMAT(/,/,&
      1X,’    IERR AT ZARSJD (STORAGE TRANSFORM CSR->JAD) = ’,I6)
 1500 FORMAT(/,/,1X,’   ** OUTPUT **’,/,/,5X,’IERR = ’,I8)
 2300 FORMAT(/,1X,2(5X, A11, 20X),/)
 2400 FORMAT(1X,’  ’,2(2X, 1PD14.7, 18X))
 2500 FORMAT(1X,’  ’,2(’(’, F14.8, ’,’, F14.8, ’)’, 3X))
      END
```

(d) Output results

```
 ***  ZCHJSS  ***

   ** INPUT PARAMETER **
     N =     4
     NA =   10
     M =     2

   ** INPUT MATRIX A **
     (-2.28,  0.00)    ( 1.78, -2.03)    ( 2.26,  0.10)    (-0.12,  2.53)
                       (-1.12,  0.00)    ( 0.01,  0.43)    (-1.07,  0.86)
                                         (-0.37,  0.00)    ( 2.31, -0.92)
                                                           (-0.73,  0.00)

   ** INPUT INDEX JA **
  JA( 1) =   1   JA( 2) =   2   JA( 3) =   3   JA( 4) =   4   JA( 5) =   2
  JA( 6) =   3   JA( 7) =   4   JA( 8) =   3   JA( 9) =   4   JA(10) =   4

   ** INPUT INDEX IA **
  IA( 1) =   1   IA( 2) =   5   IA( 3) =   8   IA( 4) =  10   IA( 5) =  11


    IERR AT ZARSJD (STORAGE TRANSFORM CSR->JAD) =      0


   ** OUTPUT **

    IERR =        0

     EIGENVALUE                         EIGENVALUE

    -6.0001855D+00                    -3.0030337D+00

     EIGENVECTOR                        EIGENVECTOR

  (   -0.69071106,   -0.23593286)   (   -0.22312724,    0.13258141)
  (    0.09074333,    0.24877543)   (    0.72843164,    0.05737392)
  (    0.34833130,    0.26867067)   (   -0.33188966,    0.02190003)
  (   -0.03041543,   -0.45020731)   (    0.50799388,    0.17333161)

     RESIDUAL                           RESIDUAL

     2.5845464D-15                     7.5446512D-16
```

4.13 GENERALIZED EIGENVALUE PROBLEM FOR A REAL MATRIX (TWO-DIMENSIONAL ARRAY TYPE)

4.13.1 DCGGAA, RCGGAA
All Eigenvalues and All Eigenvectors of a Real Matrix (Generalized Eigenvalue Problem)

(1) **Function**

DCGGAA or RCGGAA uses the Householder method and combination shift QZ method to obtain all eigenvalues of the real matrix (two-dimensional array type) generalized eigenvalue problem $A\boldsymbol{x} = \lambda B\boldsymbol{x}$ (A, B: Real matrices) and all corresponding eigenvectors.

(2) **Usage**

Double precision:

```
  CALL DCGGAA  (A, LNA, N, B, LNB, ALFR, ALFI, BETA, VE, LNV, IERR)
```

Single precision:

```
  CALL RCGGAA  (A, LNA, N, B, LNB, ALFR, ALFI, BETA, VE, LNV, IERR)
```

(3) **Arguments**

D:Double precision real   Z:Double precision complex   I: { INTEGER(4) as for 32bit Integer }
R:Single precision real   C:Single precision complex   { INTEGER(8) as for 64bit Integer }

No.	Argument	Type	Size	Input/ Output	Contents
1	A	D R	LNA, N	Input	Real matrix A (two-dimensional array type).
				Output	Input-time contents are not retained.
2	LNA	I	1	Input	Adjustable dimension of array A.
3	N	I	1	Input	Order of matrices A and B.
4	B	D R	LNB, N	Input	Real matrix B (two-dimensional array type).
				Output	Input-time contents are not retained.
5	LNB	I	1	Input	Adjustable dimension of array B.
6	ALFR	D R	N	Output	Real parts of the diagonal components of matrix A after transformation (See Note (a)).
7	ALFI	D R	N	Output	Imaginary parts of the diagonal components of matrix A after transformation (See Note (a)).
8	BETA	D R	N	Output	Diagonal components of matrix B after transformation (See Note (a)).
9	VE	D R	LNV, N	Output	Eigenvectors (See Notes (b) and (c)).

No.	Argument	Type	Size	Input/ Output	Contents
10	LNV	I	1	Input	Adjustable dimension of array VE.
11	IERR	I	1	Output	Error indicator

(4) **Restrictions**

(a) $0 < \text{N} \leq \text{LNA}, \text{LNB}, \text{LNV}$

(5) **Error indicator**

IERR value	Meaning	Processing
0	Normal termination.	
1000	N was equal to 1.	$\text{ALFR}(1) \leftarrow \text{sign}(\text{B}(1,1)) \times \text{A}(1,1)$, $\text{ALFI}(1) \leftarrow 0.0$, $\text{BETA}(1) \leftarrow \lvert\text{B}(1,1)\rvert$ and $\text{VE}(1,1) \leftarrow 1.0$ are performed.
2000	The array BETA contains 0.0.	Processing continues. (See Note (b).)
3000	Restriction (a) was not satisfied.	Processing is aborted.
$5000 + i$	The sequence did not converge in the step where the eigenvalue is obtained. $(1 \leq i \leq \text{N})$	The $(i+1)$-th through N-th elements of ALFR, ALFI and BETA are obtained. No eigenvectors is obtained.

(6) **Notes**

(a) Eigenvalues are obtained in decreasing order of their subscript values and stored in arrays ALFR, ALFI and BETA. If the j-th element of ALFR, ALFI and BETA are $\alpha_j, \alpha'_j, \beta_j$, then the eigenvalues are represented by the following formula:

(j-th eigenvalues)$= \dfrac{\alpha_j + \alpha'_j i}{\beta_j}$ (i: imaginary unit)

If the j-th eigenvalue is a real number, the 0.0 is stored in α'_j. In addition, if the j-th eigenvalue is a complex number, its conjugate complex eigenvalue is stored in the $(j+1)$-th element.
However, $\alpha'_j > 0$, $\alpha'_{j+1} < 0$ and β_j always is positive (See Figure 4−2).

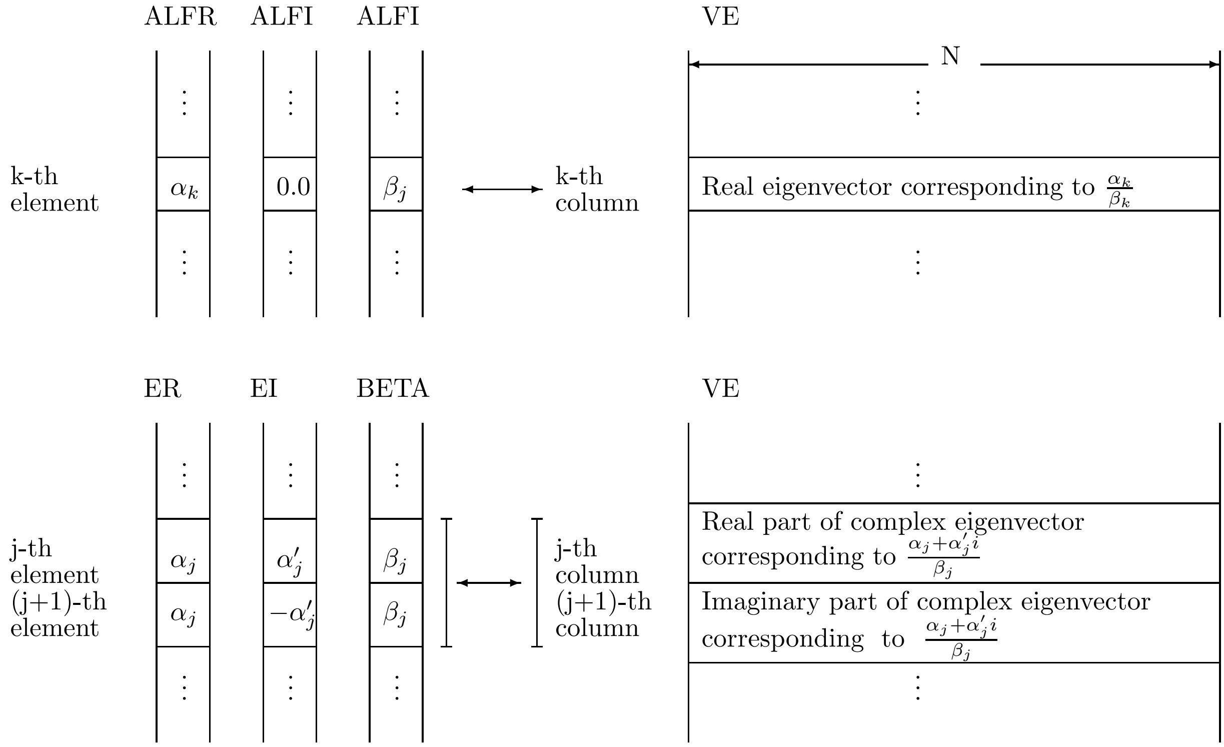
(b) IERR = 2000 indicates that the eigenvalue corresponding to $\beta_j = 0$ is an extremely large value. Note that a division by zero will occur at this time if the eigenvalue is obtained by using the formula $(\alpha_j + \alpha'_{ji})/\beta_j$.

(c) Eigenvectors corresponding to obtained eigenvalues are stored as shown in Figure 4−2 in columns of array VE. That is, if the j-th eigenvalue is real, then the eigenvector corresponding to it is stored in the j-th column of array VE. In addition, if the j-th and $(j+1)$-th eigenvalues are a pair of complex conjugate eigenvalues, then the real and imaginary parts of the complex eigenvector corresponding to the j-th element eigenvalue are stored in the j-th and $(j+1)$-th columns respectively of array VE. The conjugate vector of this complex eigenvector becomes the eigenvector corresponding to the $(j \times 1)$-th eigenvalue.

(d) An eigenvectors is normalized so that the maximum absolute value of each element is 1.0.

(e) If eigenvectors are not required, use 4.13.2 $\begin{Bmatrix} \text{DCGGAN} \\ \text{RCGGAN} \end{Bmatrix}$.

Figure 4−2 Eigenvalues and Eigenvectors Storage Format


ALFR
ALFI
ALFI
VE
N
k-th element
α_k
0.0
β_j
k-th column
Real eigenvector corresponding to $\frac{\alpha_k}{\beta_k}$
ER
EI
BETA
VE
j-th element
(j+1)-th element
α_j
α'_j
β_j
α_j
$-\alpha'_j$
β_j
j-th column
(j+1)-th column
Real part of complex eigenvector corresponding to $\frac{\alpha_j+\alpha'_j i}{\beta_j}$
Imaginary part of complex eigenvector corresponding to $\frac{\alpha_j+\alpha'_j i}{\beta_j}$

Remarks

a. $\alpha'_j > 0, \beta_j > 0$

(7) **Example**

(a) Problem

Obtain all eigenvalues of $A\boldsymbol{x} = \lambda B\boldsymbol{x}$ and their corresponding eigenvectors, where matrices A and B are as follows:

$$A = \begin{bmatrix} 50 & -60 & 50 & -27 & 6 & 6 \\ 38 & -28 & 27 & -17 & 5 & 5 \\ 27 & -17 & 27 & -17 & 5 & 5 \\ 27 & -28 & 38 & -17 & 5 & 5 \\ 27 & -28 & 27 & -17 & 16 & 5 \\ 27 & -28 & 27 & -17 & 5 & 16 \end{bmatrix}$$

$$B = \begin{bmatrix} 16 & 5 & 4 & 3 & -2 & 1 \\ 5 & 16 & 5 & 4 & -6 & 2 \\ 4 & 5 & 16 & 5 & -6 & 3 \\ 3 & 4 & 5 & 16 & -6 & 4 \\ 2 & 3 & 4 & 5 & -6 & 16 \\ 1 & 6 & 6 & 6 & -5 & 6 \end{bmatrix}$$

(b) Input data

Matrix A, LNA=11, N=6, matrix B, LNB=11 and LNV=11.

(c) Main program

```
      PROGRAM BCGGAA
! *** EXAMPLE OF DCGGAA ***
      IMPLICIT REAL(8)(A-H,O-Z)
      PARAMETER ( ZERO = 0.0D0 )
      PARAMETER (LNA=11, LNB=11, LNV=11)
      DIMENSION A(LNA,LNA), B(LNB,LNB), VE(LNV,LNV),&
                ALFR(LNA), ALFI(LNA), BETA(LNA)
!
      READ(5,*) N
      DO 10 J=1, N
         READ(5,*) (A(J,I), I=1, N)
   10 CONTINUE
      DO 20 J=1, N
         READ(5,*) (B(J,I), I=1, N)
   20 CONTINUE
!
      WRITE(6,1000) N
      WRITE(6,1100) 'A'
      DO 30 J=1, N
         WRITE(6,1200) (A(J,I), I=1, N)
   30 CONTINUE
      WRITE(6,1100) 'B'
      DO 40 J=1, N
         WRITE(6,1200) (B(J,I), I=1, N)
   40 CONTINUE
!
      CALL DCGGAA(A,LNA,N,B,LNB,ALFR,ALFI,BETA,VE,LNV,IERR)
!
      WRITE(6,1300) IERR
!
      DO 100 J=1, N-1, 2
         WRITE(6,1400) 'EIGENVALUE ', 'EIGENVALUE '
         WRITE(6,1500) 'ALFR', ALFR(J), 'ALFR', ALFR(J+1)
         WRITE(6,1500) 'ALFI', ALFI(J), 'ALFI', ALFI(J+1)
         WRITE(6,1500) 'BETA', BETA(J), 'BETA', BETA(J+1)
         WRITE(6,1400) 'EIGENVECTOR', 'EIGENVECTOR'
         IF(ALFI(J).EQ.ZERO) THEN
            IF(ALFI(J+1).EQ.ZERO) THEN
               DO 50 I=1, N
                  WRITE(6,1600) VE(I,J), ZERO, VE(I,J+1), ZERO
   50          CONTINUE
            ELSE
               DO 60 I=1, N
                  WRITE(6,1600) VE(I,J), ZERO, VE(I,J+1), VE(I,J+2)
   60          CONTINUE
            ENDIF
         ELSE
            IF(ALFI(J+1).EQ.ZERO) THEN
               DO 70 I=1, N
                  WRITE(6,1600) VE(I,J-1),-VE(I,J), VE(I,J+1), ZERO
   70          CONTINUE
            ELSE
               IF(ALFI(J).LT.ZERO) THEN
               DO 80 I=1, N
                  WRITE(6,1600) VE(I,J-1),-VE(I,J), VE(I,J+1), VE(I,J+2)
   80          CONTINUE
               ELSE
               DO 90 I=1, N
                  WRITE(6,1600) VE(I,J), VE(I,J+1), VE(I,J),-VE(I,J+1)
   90          CONTINUE
               ENDIF
            ENDIF
         ENDIF
  100 CONTINUE
      IF(MOD(N,2).NE.0) THEN
         WRITE(6,1400) 'EIGENVALUE '
         WRITE(6,1500) 'ALFR', ALFR(N)
         WRITE(6,1500) 'ALFI', ALFI(N)
         WRITE(6,1500) 'BETA', BETA(N)
         WRITE(6,1400) 'EIGENVECTOR'
         IF(ALFI(N).EQ.ZERO) THEN
            DO 110 I=1, N
               WRITE(6,1600) VE(I,N), ZERO
  110       CONTINUE
         ELSE
            DO 120 I=1, N
               WRITE(6,1600) VE(I,N-1),-VE(I,N)
  120       CONTINUE
         ENDIF
      ENDIF
      STOP
!
 1000 FORMAT(' ',/,/,&
             ' ***  DCGGAA  ***',/,/,&
             '  **  INPUT  **',/,/,&
             '      N = ', I2)
 1100 FORMAT(' ',/,&
             '      INPUT MATRIX ',A1,/)
 1200 FORMAT(7X, 10(F7.1))
 1300 FORMAT(' ',/,/,&
             '  **  OUTPUT **',/,/,&
             '      IERR = ', I4)
```

```
1400 FORMAT(' ',/,14X,2(A11,22X))
1500 FORMAT(' ',2(7X,A4,' = ',1PD14.7,5X))
1600 FORMAT(' ',2(5X,F12.8,' ,',F12.8,2X))
     END
```

(d) Output results

```
***  DCGGAA  ***

 **  INPUT  **

     N =  6

     INPUT MATRIX A

         50.0  -60.0   50.0  -27.0    6.0    6.0
         38.0  -28.0   27.0  -17.0    5.0    5.0
         27.0  -17.0   27.0  -17.0    5.0    5.0
         27.0  -28.0   38.0  -17.0    5.0    5.0
         27.0  -28.0   27.0  -17.0   16.0    5.0
         27.0  -28.0   27.0  -17.0    5.0   16.0

     INPUT MATRIX B

         16.0    5.0    4.0    3.0   -2.0    1.0
          5.0   16.0    5.0    4.0   -6.0    2.0
          4.0    5.0   16.0    5.0   -6.0    3.0
          3.0    4.0    5.0   16.0   -6.0    4.0
          2.0    3.0    4.0    5.0   -6.0   16.0
          1.0    6.0    6.0    6.0   -5.0    6.0


 **  OUTPUT **

     IERR =    0

             EIGENVALUE                       EIGENVALUE
       ALFR = -1.1634546D+01           ALFR =  1.1837890D+01
       ALFI =  0.0000000D+00           ALFI =  0.0000000D+00
       BETA =  9.3588312D-01           BETA =  3.9038739D+00

             EIGENVECTOR                      EIGENVECTOR
      -0.02489667 ,  0.00000000        0.19987257 ,  0.00000000
       0.25251218 ,  0.00000000       -0.19189298 ,  0.00000000
       0.19443080 ,  0.00000000       -0.24240591 ,  0.00000000
       0.20492111 ,  0.00000000       -0.21760871 ,  0.00000000
       1.00000000 ,  0.00000000       -1.00000000 ,  0.00000000
       0.16361487 ,  0.00000000       -0.44802812 ,  0.00000000

             EIGENVALUE                       EIGENVALUE
       ALFR =  6.6131567D+00           ALFR =  9.3466830D+00
       ALFI =  1.5653458D+01           ALFI = -2.2123763D+01
       BETA =  1.4195630D+01           BETA =  2.0063347D+01

             EIGENVECTOR                      EIGENVECTOR
      -0.90275293 , -0.16929219       -0.90275293 ,  0.16929219
      -0.61708834 ,  0.78689388       -0.61708834 , -0.78689388
       0.12677136 ,  0.78675507        0.12677136 , -0.78675507
       0.25033712 ,  0.21412357        0.25033712 , -0.21412357
      -0.41100766 ,  0.54117624       -0.41100766 , -0.54117624
      -0.22010716 ,  0.61947141       -0.22010716 , -0.61947141

             EIGENVALUE                       EIGENVALUE
       ALFR =  4.8318024D+00           ALFR =  5.7494668D+00
       ALFI =  7.8056571D+00           ALFI = -9.2881212D+00
       BETA =  1.0797432D+01           BETA =  1.2848100D+01

             EIGENVECTOR                      EIGENVECTOR
       0.46429690 , -0.29190301        0.46429690 ,  0.29190301
       0.06840004 , -0.86309624        0.06840004 ,  0.86309624
      -0.60524575 , -0.79603869       -0.60524575 ,  0.79603869
      -0.88694870 , -0.14801432       -0.88694870 ,  0.14801432
      -0.21908532 , -0.37959993       -0.21908532 ,  0.37959993
      -0.21818206 , -0.41595648       -0.21818206 ,  0.41595648
```

4.13.2 DCGGAN, RCGGAN
All Eigenvalues of a Real Matrix (Generalized Eigenvalue Problem)

(1) **Function**

DCGGAN or RCGGAN uses the Householder method and combination shift QZ method to obtain all eigenvalues of the real matrix (two-dimensional array type) generalized eigenvalue problem $A\boldsymbol{x} = \lambda B\boldsymbol{x}$ (A, B: Real matrices).

(2) **Usage**

Double precision:

CALL DCGGAN (A, LNA, N, B, LNB, ALFR, ALFI, BETA, IERR)

Single precision:

CALL RCGGAN (A, LNA, N, B, LNB, ALFR, ALFI, BETA, IERR)

(3) **Arguments**

D:Double precision real Z:Double precision complex I: $\left\{\begin{array}{l}\text{INTEGER(4) as for 32bit Integer}\\ \text{INTEGER(8) as for 64bit Integer}\end{array}\right\}$
R:Single precision real C:Single precision complex

No.	Argument	Type	Size	Input/ Output	Contents
1	A	$\left\{\begin{array}{c}\text{D}\\ \text{R}\end{array}\right\}$	LNA, N	Input	Real matrix A (two-dimensional array type).
				Output	Input-time contents are not retained.
2	LNA	I	1	Input	Adjustable dimension of array A.
3	N	I	1	Input	Order of matrices A and B.
4	B	$\left\{\begin{array}{c}\text{D}\\ \text{R}\end{array}\right\}$	LNB, N	Input	Real matrix B (two-dimensional array type).
				Output	Input-time contents are not retained.
5	LNB	I	1	Input	Adjustable dimension of array B.
6	ALFR	$\left\{\begin{array}{c}\text{D}\\ \text{R}\end{array}\right\}$	N	Output	Real parts of the diagonal components of matrix A after transformation (See Note (a)).
7	ALFI	$\left\{\begin{array}{c}\text{D}\\ \text{R}\end{array}\right\}$	N	Output	Imaginary parts of the diagonal components of matrix A after transformation (See Note (a)).
8	BETA	$\left\{\begin{array}{c}\text{D}\\ \text{R}\end{array}\right\}$	N	Output	Diagonal components of matrix B after transformation (See Note (a)).
9	IERR	I	1	Output	Error indicator

(4) **Restrictions**

(a) $0 < \mathrm{N} \leq \mathrm{LNA}, \mathrm{LNB}$

(5) **Error indicator**

IERR value	Meaning	Processing
0	Normal termination.	
1000	N was equal to 1.	ALFR(1) ← sign(B(1, 1)) × A(1, 1), ALFI(1) ← 0.0 and BETA(1) ← |B(1, 1)| are performed.
2000	The array BETA contains 0.0.	Processing continues. (See Note (b).)
3000	Restriction (a) was not satisfied.	Processing is aborted.
$5000 + i$	The sequence did not converge in the step where the eigenvalue is obtained. $(1 \leq i \leq \mathrm{N})$	The $(i + 1)$-th through N-th elements of ALFR, ALFI and BETA are obtained.

(6) **Notes**

(a) Eigenvalues are obtained in decreasing order of their subscript values and stored in arrays ALFR, ALFI and BETA. If the j-th element of ALFR, ALFI and BETA are $\alpha_j, \alpha'_j, \beta_j$, then the eigenvalues are represented by the following formula:
$(j$-th eigenvalues$) = \dfrac{\alpha_j + \alpha'_j i}{\beta_j}$ (i: imaginary unit)
If the j-th eigenvalue is a real number, the 0.0 is stored in α'_j. In addition, if the j-th eigenvalue is a complex number, its conjugate complex eigenvalue is stored in the $(j + 1)$-th element.
However, $\alpha'_j > 0$, $\alpha'_{j+1} < 0$ and β_j always is positive. (See Figure 4−2).

(b) IERR = 2000 indicates that the eigenvalue corresponding to $\beta_j = 0$ is an extremely large value. Note that a division by zero will occur at this time if the eigenvalue is obtained by using the formula $(\alpha_j + \alpha'_{ji})/\beta_j$.

4.14 GENERALIZED EIGENVALUE PROBLEM ($Ax = \lambda Bx$) FOR A REAL SYMMETRIC MATRIX (TWO-DIMENSIONAL ARRAY TYPE) (UPPER TRIANGULAR TYPE)

4.14.1 DCGSAA, RCGSAA
All Eigenvalues and All Eigenvectors of a Real Symmetric Matrix (Generalized Eigenvalue Problem $Ax = \lambda Bx$, B: Positive)

(1) **Function**

DCGSAA or RCGSAA uses the Cholesky method to transform the real symmetric matrix (two-dimensional array type) (upper triangular type) generalized eigenvalue problem $A\boldsymbol{x} = \lambda B\boldsymbol{x}$ (A: Real symmetric matrix, B: Positive symmetric matrix) to a standard eigenvalue problem and uses the Householder method and QR method to obtain all eigenvalues and corresponding all eigenvectors.

(2) **Usage**

Double precision:
  CALL DCGSAA (A, LNA, N, B, LNB, E, W1, IERR)
Single precision:
  CALL RCGSAA (A, LNA, N, B, LNB, E, W1, IERR)

(3) **Arguments**

D:Double precision real Z:Double precision complex I: $\left\{\begin{matrix}\text{INTEGER(4) as for 32bit Integer}\\ \text{INTEGER(8) as for 64bit Integer}\end{matrix}\right\}$
R:Single precision real C:Single precision complex

No.	Argument	Type	Size	Input/ Output	Contents
1	A	$\left\{\begin{matrix}D\\R\end{matrix}\right\}$	LNA, N	Input	Real symmetric matrix A (two-dimensional array type) (upper triangular type)
				Output	Eigenvectors (column vector) corresponding to each eigenvalue
2	LNA	I	1	Input	Adjustable dimension of array A
3	N	I	1	Input	Order of matrix A and B
4	B	$\left\{\begin{matrix}D\\R\end{matrix}\right\}$	LNB, N	Input	Politics symmetric matrix B (two-dimensional array type) (upper triangular type)
				Output	Input-time contents are not retained.
5	LNB	I	1	Input	Adjustable dimension of array B
6	E	$\left\{\begin{matrix}D\\R\end{matrix}\right\}$	N	Output	Eigenvalues
7	W1	$\left\{\begin{matrix}D\\R\end{matrix}\right\}$	2 × N	Work	Work area
8	IERR	I	1	Output	Error indicator

(4) **Restrictions**

(a) $0 < \mathrm{N} \leq \mathrm{LNA}, \mathrm{LNB}$

(5) **Error indicator**

IERR value	Meaning	Processing
0	Normal termination.	
1000	N was equal to 1.	$\mathrm{E}(1) \leftarrow \mathrm{A}(1,1)/\mathrm{B}(1,1)$ and $\mathrm{A}(1,1) \leftarrow 1.0/\sqrt{\mathrm{B}(1,1)}$ are performed.
2100	B has a diagonal element very close to zero.	Some eigenvectors may be obtained with low precision, and processing continues.
3000	Restriction (a) was not satisfied.	Processing is aborted.
4000	B was not positive definite.	
$5000 + i$	The sequence did not converge in the step where the eigenvalue is obtained. $(1 \leq i \leq \mathrm{N})$	Eigenvalues obtained by this time are entered in $\mathrm{E}(1), \cdots, \mathrm{E}(\mathrm{i}-1)$ (However, the order is irregular). No eigenvector is obtained at this time.

(6) **Notes**

(a) Data should be stored only in the upper triangular portions of arrays A and B.

(b) Eigenvalues are stored in ascending order.

(c) Eigenvectors $\boldsymbol{v}_i$ are an orthonormal set so that $\boldsymbol{v}_j^T B \boldsymbol{v}_k = \delta_{j,k}$

(d) If eigenvectors are not required, use 4.14.2 $\left\{\begin{matrix}\mathrm{DCGSAN}\\ \mathrm{RCGSAN}\end{matrix}\right\}$.

(7) **Example**

(a) Problem
Obtain all eigenvalues of $A\boldsymbol{x} = \lambda B\boldsymbol{x}$ and their corresponding eigenvectors, where matrices A and B are as follows:

$$A = \begin{bmatrix} 2 & 1 & 1 & 2 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 2 & 2 \\ 2 & 1 & 2 & 4 \end{bmatrix}$$

$$B = \begin{bmatrix} 153 & 31 & 58 & -58 \\ 31 & 153 & -53 & 58 \\ 58 & -58 & 153 & 31 \\ -58 & 58 & 31 & 153 \end{bmatrix}$$

(b) Input data
Matrix A, LNA=11, N=4, matrix B and LNB=11.

(c) Main program

```
      PROGRAM BCGSAA
! *** EXAMPLE OF DCGSAA ***
      IMPLICIT REAL(8) (A-H,O-Z)
      PARAMETER ( LNA = 11, LNB = 11 )
      DIMENSION A(LNA,LNA), B(LNB,LNB), E(LNA), W1(2*LNA)
!
      READ(5,*) N
      DO 10 J=1, N
         READ(5,*) (A(J,I), I=J, N)
   10 CONTINUE
      DO 20 J=1, N
         READ(5,*) (B(J,I), I=J, N)
   20 CONTINUE
!
      WRITE(6,1000) N
      WRITE(6,1100) 'A'
      DO 30 J=1, N
         WRITE(6,1200) (A(I,J), I=1, J-1), (A(J,I), I=J, N)
   30 CONTINUE
      WRITE(6,1100) 'B'
      DO 40 J=1, N
         WRITE(6,1200) (B(I,J), I=1, J-1), (B(J,I), I=J, N)
   40 CONTINUE
!
      CALL DCGSAA(A,LNA,N,B,LNB,E,W1,IERR)
!
      WRITE(6,1300) IERR
!
      DO 60 K=1, N-3, 4
         WRITE(6,1400) ('EIGENVALUE ', I=1, 4)
         WRITE(6,1500) (E(I), I=K, K+3)
         WRITE(6,1400) ('EIGENVECTOR', I=1, 4)
         DO 50 J=1, N
            WRITE(6,1500) (A(J,I), I=K, K+3)
   50    CONTINUE
   60 CONTINUE
      IF(MOD(N,4).NE.0) THEN
         WRITE(6,1400) ('EIGENVALUE ', I=N/4*4+1, N)
         WRITE(6,1500) (E(I), I=N/4*4+1, N)
         WRITE(6,1400) ('EIGENVECTOR', I=N/4*4+1, N)
         DO 70 J=1, N
            WRITE(6,1500) (A(J,I), I=N/4*4+1, N)
   70    CONTINUE
      ENDIF
      STOP
!
 1000 FORMAT(' ',/,/,&
             ' ***  DCGSAA  ***',/,/,&
             '  **  INPUT  **',/,/,&
             '      N = ', I2)
 1100 FORMAT(' ',/,&
             '      INPUT MATRIX ',A1,/)
 1200 FORMAT(7X, 11(F8.1))
 1300 FORMAT(' ',/,/,&
             '  **  OUTPUT  **',/,/,&
             '      IERR = ', I4)
 1400 FORMAT(' ',/,1X, 4(5X, A11, 2X))
 1500 FORMAT(' ', 4(2X, 1PD14.7, 2X))
      END
```

(d) Output results

```
***  DCGSAA  ***

 **  INPUT  **

     N =  4

     INPUT MATRIX A

           2.0     1.0     1.0     2.0
           1.0     1.0     1.0     1.0
           1.0     1.0     2.0     2.0
           2.0     1.0     2.0     4.0

     INPUT MATRIX B

         153.0    31.0    58.0   -58.0
          31.0   153.0   -58.0    58.0
          58.0   -58.0   153.0    31.0
         -58.0    58.0    31.0   153.0


 **  OUTPUT  **

     IERR =    0

     EIGENVALUE        EIGENVALUE        EIGENVALUE        EIGENVALUE
  6.4779811D-04     5.3688291D-03     2.7447086D-02     2.1668091D-01

     EIGENVECTOR       EIGENVECTOR       EIGENVECTOR       EIGENVECTOR
  2.9405960D-02     4.9839235D-02    -1.6115522D-02     2.0451432D-01
 -4.6877602D-02     3.7709049D-02     6.8634504D-02    -1.9262477D-01
  3.1083549D-02    -1.9357438D-02     8.5979167D-02    -1.9157548D-01
 -1.9570768D-02    -3.3245027D-02     1.4513123D-03     2.0962854D-01
```

4.14.2 DCGSAN, RCGSAN All Eigenvalues of a Real Symmetric Matrix (Generalized Eigenvalue Problem $Ax = \lambda Bx$, B: Positive)

(1) **Function**

DCGSAN or RCGSAN uses the Cholesky method to transform the real symmetric matrix (two-dimensional array type) (upper triangular type) generalized eigenvalue problem $A\boldsymbol{x} = \lambda B\boldsymbol{x}$ (A: Real symmetric matrix, B: Positive symmetric matrix) to a standard eigenvalue problem and uses the Householder method and QR method to obtain all eigenvalues.

(2) **Usage**

Double precision:
CALL DCGSAN (A, LNA, N, B, LNB, E, W1, IERR)
Single precision:
CALL RCGSAN (A, LNA, N, B, LNB, E, W1, IERR)

(3) **Arguments**

D:Double precision real Z:Double precision complex
R:Single precision real C:Single precision complex
I: $\left\{\begin{array}{l}\text{INTEGER(4) as for 32bit Integer}\\ \text{INTEGER(8) as for 64bit Integer}\end{array}\right\}$

No.	Argument	Type	Size	Input/ Output	Contents
1	A	$\begin{Bmatrix}\text{D}\\ \text{R}\end{Bmatrix}$	LNA, N	Input	Real symmetric matrix A (two-dimensional array type) (upper triangular type)
				Output	Input-time contents are not retained.
2	LNA	I	1	Input	Adjustable dimension of array A
3	N	I	1	Input	Order of matrix A and B
4	B	$\begin{Bmatrix}\text{D}\\ \text{R}\end{Bmatrix}$	LNB, N	Input	Positive symmetric matrix B (two-dimensional array type) (upper triangular type)
				Output	Input-time contents are not retained.
5	LNB	I	1	Input	Adjustable dimension of array B
6	E	$\begin{Bmatrix}\text{D}\\ \text{R}\end{Bmatrix}$	N	Output	Eigenvalues
7	W1	$\begin{Bmatrix}\text{D}\\ \text{R}\end{Bmatrix}$	N	Work	Work area
8	IERR	I	1	Output	Error indicator

(4) **Restrictions**

(a) $0 < \text{N} \leq \text{LNA}, \text{LNB}$

(5) **Error indicator**

IERR value	Meaning	Processing
0	Normal termination.	
1000	N was equal to 1.	$\mathrm{E}(1) \leftarrow \mathrm{A}(1,1)/\mathrm{B}(1,1)$ is performed.
2100	B has a diagonal element very close to zero.	Processing continues.
3000	Restriction (a) was not satisfied.	Processing is aborted.
4000	B was not positive definite.	
$5000+i$	The sequence did not converge in the step where the eigenvalue is obtained. $(1 \leq i \leq \mathrm{N})$	Eigenvalues obtained by this time are entered in $\mathrm{E}(1), \cdots, \mathrm{E}(\mathrm{i}-1)$ (However, the order is irregular).

(6) **Notes**

(a) Data should be stored only in the upper triangular portions of arrays A and B.

(b) Eigenvalues are stored in ascending order.

4.14.3 DCGSSS, RCGSSS Eigenvalues and Eigenvectors of a Real Symmetric Matrix (Generalized Eigenvalue Problem $Ax = \lambda Bx$, B: Positive)

(1) **Function**

DCGSSS or RCGSSS uses the Cholesky method to transform the real symmetric matrix (two-dimensional array type) (upper triangular type) generalized eigenvalue problem $A\boldsymbol{x} = \lambda B\boldsymbol{x}$ (A: Real symmetric matrix, B: Positive symmetric matrix) to a standard eigenvalue problem and uses the Householder method and the root-free QR method or Bisection method to obtain the m largest eigenvalues or m smallest eigenvalues, and uses the reverse iterative method to obtain the eigenvectors.

(2) **Usage**

Double precision:
CALL DCGSSS (A, LNA, N, B, LNB, EPS, E, M, VE, LNV, ISW, IW1, W1, IERR)
Single precision:
CALL RCGSSS (A, LNA, N, B, LNB, EPS, E, M, VE, LNV, ISW, IW1, W1, IERR)

(3) **Arguments**

D:Double precision real Z:Double precision complex
R:Single precision real C:Single precision complex
I: $\left\{\begin{array}{l}\text{INTEGER(4) as for 32bit Integer}\\\text{INTEGER(8) as for 64bit Integer}\end{array}\right\}$

No.	Argument	Type	Size	Input/ Output	Contents
1	A	$\left\{\begin{matrix}D\\R\end{matrix}\right\}$	LNA, N	Input	Real symmetric matrix A (two-dimensional array type) (upper triangular type)
				Output	Input-time contents are not retained.
2	LNA	I	1	Input	Adjustable dimension of array A
3	N	I	1	Input	Order of matrix A and B
4	B	$\left\{\begin{matrix}D\\R\end{matrix}\right\}$	LNB, N	Input	Positive symmetric matrix B (two-dimensional array type) (upper triangular type)
				Output	The strict upper triangular portion is not retained.
5	LNB	I	1	Input	Adjustable dimension of array B
6	EPS	$\left\{\begin{matrix}D\\R\end{matrix}\right\}$	1	Input	Parameter that assigns an upper limit to the absolute error for use in the eigenvalue convergence test. (See Note (d))
7	E	$\left\{\begin{matrix}D\\R\end{matrix}\right\}$	M	Output	Eigenvalues
8	M	I	1	Input	The number of m of eigenvalues to be obtained.
9	VE	$\left\{\begin{matrix}D\\R\end{matrix}\right\}$	LNV, M	Output	Eigenvectors (column vector) corresponding to each eigenvalue

No.	Argument	Type	Size	Input/ Output	Contents
10	LNV	I	1	Input	Adjustable dimension of array VE
11	ISW	I	1	Input	Processing switch ISW≥0: Obtain M eigenvalues from the largest one. ISW<0: Obtain M eigenvalues from the smallest one.
12	IW1	I	M	Output	Eigenvector flag (See Note (a))
13	W1	$\begin{Bmatrix} \mathrm{D} \\ \mathrm{R} \end{Bmatrix}$	9 × N	Work	Work area
14	IERR	I	1	Output	Error indicator

(4) **Restrictions**

(a) $0 < \mathrm{N} \leq \mathrm{LNA}, \mathrm{LNB}, \mathrm{LNV}$

(b) $0 < \mathrm{M} \leq \mathrm{N}$

(5) **Error indicator**

IERR value	Meaning	Processing
0	Normal termination.	
1000	N was equal to 1.	$\mathrm{E}(1) \leftarrow \mathrm{A}(1,1)/\mathrm{B}(1,1)$ and $\mathrm{VE}(1,1) \leftarrow 1.0/\sqrt{\mathrm{B}(1,1)}$ are performed.
2000	The maximum number of iterations was exceeded by the inverse iterations for obtaining eigenvectors.	Some eigenvectors are obtained with low precision, and processing continues. (See Note (e).)
2100	B has a diagonal element very close to zero.	Some eigenvectors may be obtained with low precision, and processing continues.
3000	Restriction (a) or (b) was not satisfied.	Processing is aborted.
4000	B was not positive definite.	

(6) **Notes**

(a) Data should be stored only in the upper triangular portions of arrays A and B.

(b) If ISW≥0, the eigenvalues are stored in descending order. If ISW<0, they are stored in ascending order.

(c) Eigenvalue calculations are appropriately divided up between the root-free QR method and Bisection method internally.

(d) If EPS ≤ 0, the optimum value is automatically set internally. Normally, a negative value should be set so that this value will be set automatically. EPS is used to obtain eigenvalues by using the Bisection method.

(e) If the maximum number of iterations is exceeded when using the inverse iteration method (IERR = 2000 is output), the following processing is performed.

If IW1(i) = 0: The i-th eigenvector calculation is normally terminated.

If IW1(i) $\neq 0$: The convergence condition is not satisfied for the i-th eigenvector calculation, and the eigenvector precision is low. In this case, the iteration count is set for IW1(i).

If processing is normally terminated (IERR = 0 is output), IW1(i) = 0 is set.

(f) Eigenvectors $\boldsymbol{v}_i$ are an orthonormal set so that $\boldsymbol{v}_j^T B \boldsymbol{v}_k = \delta_{j,k}$

(g) If eigenvectors are not required, use 4.14.4 $\left\{\begin{matrix}\text{DCGSSN}\\ \text{RCGSSN}\end{matrix}\right\}$.

(7) **Example**

(a) ProblemObtain all eigenvalues of $A\boldsymbol{x} = \lambda B\boldsymbol{x}$ and their corresponding eigenvectors, where matrices A and B are as follows:

$$A = \begin{bmatrix} 611 & 196 & -192 & 407 & -8 & -52 & -49 & 29 \\ 196 & 899 & 113 & -192 & -71 & -43 & -8 & -44 \\ -192 & 113 & 899 & 196 & 61 & 49 & 8 & 52 \\ 407 & -192 & 196 & 611 & 8 & 44 & 59 & -23 \\ -8 & -71 & 61 & 8 & 411 & -599 & 208 & 208 \\ -52 & -43 & 49 & 44 & -599 & 411 & 208 & 208 \\ -49 & -8 & 8 & 59 & 208 & 208 & 99 & -911 \\ 29 & -44 & 52 & -23 & 208 & 208 & -911 & 99 \end{bmatrix}$$

$$B = \begin{bmatrix} 170 & 18 & 33 & -21 & -17 & 13 & 25 & -36 \\ 18 & 171 & -21 & 22 & 13 & -17 & -36 & 25 \\ 33 & -21 & 171 & 18 & 25 & -36 & -17 & 13 \\ -21 & 22 & 18 & 171 & -36 & 25 & 13 & -17 \\ -17 & 13 & 25 & -36 & 171 & 18 & 33 & -21 \\ 13 & -17 & -36 & 25 & 18 & 171 & -21 & -3 \\ 25 & -36 & -17 & 13 & 33 & -21 & 171 & 18 \\ -36 & 25 & 13 & -17 & -21 & -3 & 18 & 171 \end{bmatrix}$$

(b) Input data

Matrix A, LNA=11, N=8, matrix B, LNB=11, EPS=-1.0, M=2, LNV=10 and ISW=-1.

(c) Main program

```
      PROGRAM BCGSSS
! *** EXAMPLE OF DCGSSS ***
      IMPLICIT REAL(8) (A-H,O-Z)
      PARAMETER ( LNA = 11, LNB = 11, LNV = 10 )
      DIMENSION A(LNA,LNA), B(LNB,LNB), E(LNA), VE(LNV,LNV),&
              IW1(LNA), W1(9*LNA)
!
      READ(5,*) N, M
      DO 10 J=1, N
         READ(5,*) (A(J,I), I=J, N)
   10 CONTINUE
      DO 20 J=1, N
         READ(5,*) (B(J,I), I=J, N)
   20 CONTINUE
!
      WRITE(6,1000) N, M
      WRITE(6,1100) 'A'
      DO 30 J=1, N
         WRITE(6,1200) (A(I,J), I=1, J-1), (A(J,I), I=J, N)
   30 CONTINUE
      WRITE(6,1100) 'B'
      DO 40 J=1, N
         WRITE(6,1200) (B(I,J), I=1, J-1), (B(J,I), I=J, N)
   40 CONTINUE
!
      ISW = -1
      EPS = -1.0D0
!
      CALL DCGSSS(A,LNA,N,B,LNB,EPS,E,M,VE,LNV,ISW,IW1,W1,IERR)
!
```

```
      WRITE(6,1300) IERR
!
      DO 60 K=1, M-3, 4
         WRITE(6,1400) ('EIGENVALUE ', I=1, 4)
         WRITE(6,1500) (E(I), I=K, K+3)
         WRITE(6,1400) ('EIGENVECTOR', I=1, 4)
         DO 50 J=1, N
            WRITE(6,1500) (VE(J,I), I=K, K+3)
   50    CONTINUE
   60 CONTINUE
      IF(MOD(M,4).NE.0) THEN
         WRITE(6,1400) ('EIGENVALUE ', I=M/4*4+1, M)
         WRITE(6,1500) (E(I), I=M/4*4+1, M)
         WRITE(6,1400) ('EIGENVECTOR', I=M/4*4+1, M)
         DO 70 J=1, N
            WRITE(6,1500) (VE(J,I), I=M/4*4+1, M)
   70    CONTINUE
      ENDIF
      STOP
!
 1000 FORMAT(' ',/,/,&
             ' ***  DCGSSS  ***',/,/,&
             '  **  INPUT  **',/,/,&
             '      N = ', I2,/,/,&
             '      M = ', I2)
 1100 FORMAT(' ',/,&
             '      INPUT MATRIX ',A1,/)
 1200 FORMAT(7X, 11(F8.1))
 1300 FORMAT(' ',/,/,&
             '  **  OUTPUT  **',/,/,&
             '      IERR = ', I4)
 1400 FORMAT(' ',/,1X, 4(5X, A11, 2X))
 1500 FORMAT(' ', 4(2X, 1PD14.7, 2X))
      END
```

(d) Output results

```
***  DCGSSS  ***

 **  INPUT  **

     N =  8

     M =  2

     INPUT MATRIX A

          611.0   196.0  -192.0   407.0    -8.0   -52.0   -49.0    29.0
          196.0   899.0   113.0  -192.0   -71.0   -43.0    -8.0   -44.0
         -192.0   113.0   899.0   196.0    61.0    49.0     8.0    52.0
          407.0  -192.0   196.0   611.0     8.0    44.0    59.0   -23.0
           -8.0   -71.0    61.0     8.0   411.0  -599.0   208.0   208.0
          -52.0   -43.0    49.0    44.0  -599.0   411.0   208.0   208.0
          -49.0    -8.0     8.0    59.0   208.0   208.0    99.0  -911.0
           29.0   -44.0    52.0   -23.0   208.0   208.0  -911.0    99.0

     INPUT MATRIX B

          170.0    18.0    33.0   -21.0   -17.0    13.0    25.0   -36.0
           18.0   171.0   -21.0    22.0    13.0   -17.0   -36.0    25.0
           33.0   -21.0   171.0    18.0    25.0   -36.0   -17.0    13.0
          -21.0    22.0    18.0   171.0   -36.0    25.0    13.0   -17.0
          -17.0    13.0    25.0   -36.0   171.0    18.0    33.0   -21.0
           13.0   -17.0   -36.0    25.0    18.0   171.0   -21.0    -3.0
           25.0   -36.0   -17.0    13.0    33.0   -21.0   171.0    18.0
          -36.0    25.0    13.0   -17.0   -21.0    -3.0    18.0   171.0


 **  OUTPUT  **

     IERR =    0

     EIGENVALUE         EIGENVALUE
 -5.3020806D+00    -1.0369304D-15

     EIGENVECTOR        EIGENVECTOR
  7.8865043D-04    -3.2898475D-03
  1.4571715D-03    -6.5796950D-03
  6.2438782D-04     6.5796950D-03
 -1.6786293D-03     3.2898475D-03
 -2.4707363D-02    -4.6057865D-02
 -1.9017760D-02    -4.6057865D-02
  4.7852016D-02    -2.3028932D-02
  4.4548878D-02    -2.3028932D-02
```

4.14.4 DCGSSN, RCGSSN Eigenvalues of a Real Symmetric Matrix (Generalized Eigenvalue Problem $Ax = \lambda Bx$, B: Positive)

(1) **Function**

DCGSSN or RCGSSN uses the Cholesky method to transform the real symmetric matrix (two-dimensional array type) (upper triangular type) generalized eigenvalue problem $A\boldsymbol{x} = \lambda B\boldsymbol{x}$ (A: Real symmetric matrix, B: Positive symmetric matrix) to a standard eigenvalue problem and uses the Householder method and the root-free QR method or Bisection method to obtain the m largest eigenvalues or m smallest eigenvalues.

(2) **Usage**

Double precision:
CALL DCGSSN (A, LNA, N, B, LNB, EPS, E, M, ISW, W1, IERR)
Single precision:
CALL RCGSSN (A, LNA, N, B, LNB, EPS, E, M, ISW, W1, IERR)

(3) **Arguments**

D:Double precision real Z:Double precision complex
R:Single precision real C:Single precision complex
I: $\left\{\begin{array}{l}\text{INTEGER(4) as for 32bit Integer}\\ \text{INTEGER(8) as for 64bit Integer}\end{array}\right\}$

No.	Argument	Type	Size	Input/ Output	Contents
1	A	$\left\{\begin{array}{c}\text{D}\\ \text{R}\end{array}\right\}$	LNA, N	Input	Real symmetric matrix A (two-dimensional array type) (upper triangular type)
				Output	Input-time contents are not retained.
2	LNA	I	1	Input	Adjustable dimension of array A
3	N	I	1	Input	Order of matrix A and B
4	B	$\left\{\begin{array}{c}\text{D}\\ \text{R}\end{array}\right\}$	LNB, N	Input	Positive symmetric matrix B (two-dimensional array type) (upper triangular type)
				Output	Input-time contents are not retained.
5	LNB	I	1	Input	Adjustable dimension of array B
6	EPS	$\left\{\begin{array}{c}\text{D}\\ \text{R}\end{array}\right\}$	1	Input	Parameter that assigns an upper limit to the absolute error for use in the eigenvalue convergence test. (See Note (d))
7	E	$\left\{\begin{array}{c}\text{D}\\ \text{R}\end{array}\right\}$	M	Output	Eigenvalues
8	M	I	1	Input	The number of m of eigenvalues to be obtained.
9	ISW	I	1	Input	Processing switch ISW≥0: Obtain M eigenvalues from the largest one. ISW<0: Obtain M eigenvalues from the smallest one.

No.	Argument	Type	Size	Input/ Output	Contents
10	W1	$\begin{Bmatrix} \mathrm{D} \\ \mathrm{R} \end{Bmatrix}$	5 × N	Work	Work area
11	IERR	I	1	Output	Error indicator

(4) **Restrictions**

(a) $0 < \mathrm{N} \le \mathrm{LNA}, \mathrm{LNB}$

(b) $0 < \mathrm{M} \le \mathrm{N}$

(5) **Error indicator**

IERR value	Meaning	Processing
0	Normal termination.	
1000	N was equal to 1.	$\mathrm{E}(1) \leftarrow \mathrm{A}(1,1)/\mathrm{B}(1,1)$ is performed.
2100	B has a diagonal element very close to zero.	Processing continues.
3000	Restriction (a) or (b) was not satisfied.	Processing is aborted.
4000	B was not positive definite.	

(6) **Notes**

(a) Data should be stored only in the upper triangular portions of arrays A and B.

(b) If ISW≥0, the eigenvalues are stored in descending order. If ISW<0, they are stored in ascending order.

(c) Eigenvalue calculations are appropriately divided up between the root-free QR method and Bisection method internally.

(d) If EPS ≤ 0, the optimum value is automatically set internally. Normally, a negative value should be set so that this value will be set automatically. EPS is used to obtain eigenvalues by using the Bisection method.

4.14.5 DCGSEE, RCGSEE Eigenvalues in an Interval and Their Eigenvectors of a Real Symmetric Matrix (Interval Specified) (Generalized Eigenvalue Problem $Ax = \lambda Bx$, B: Positive)

(1) **Function**

DCGSEE or RCGSEE uses the Cholesky method to transform the real symmetric matrix (two-dimensional array type) (upper triangular type) generalized eigenvalue problem $A\boldsymbol{x} = \lambda B\boldsymbol{x}$ (A: Real symmetric matrix, B: Positive symmetric matrix) to a standard eigenvalue problem and uses the Householder method and the Bisection method to obtain the m largest eigenvalues or m smallest eigenvalues in a specified interval and uses the reverse iterative method to obtain the eigenvectors.

(2) **Usage**

Double precision:
CALL DCGSEE (A, LNA, N, B, LNB, EPS, E, M, E1, E2, VE, LNV, IW1, W1, IERR)
Single precision:
CALL RCGSEE (A, LNA, N, B, LNB, EPS, E, M, E1, E2, VE, LNV, IW1, W1, IERR)

(3) **Arguments**

D:Double precision real	Z:Double precision complex	I: $\left\{\begin{array}{l}\text{INTEGER(4) as for 32bit Integer}\\ \text{INTEGER(8) as for 64bit Integer}\end{array}\right\}$
R:Single precision real	C:Single precision complex	

No.	Argument	Type	Size	Input/ Output	Contents
1	A	$\begin{Bmatrix}\text{D}\\ \text{R}\end{Bmatrix}$	LNA, N	Input	Real symmetric matrix A (two-dimensional array type) (upper triangular type)
				Output	Input-time contents are not retained.
2	LNA	I	1	Input	Adjustable dimension of array A
3	N	I	1	Input	Order of matrix A and B
4	B	$\begin{Bmatrix}\text{D}\\ \text{R}\end{Bmatrix}$	LNB, N	Input	Positive symmetric matrix B (two-dimensional array type) (upper triangular type)
				Output	The strict upper triangular portion is not retained.
5	LNB	I	1	Input	Adjustable dimension of array B
6	EPS	$\begin{Bmatrix}\text{D}\\ \text{R}\end{Bmatrix}$	1	Input	Parameter that assigns an upper limit to the absolute error for use in the eigenvalue convergence test. (See Note (b))
7	E	$\begin{Bmatrix}\text{D}\\ \text{R}\end{Bmatrix}$	M	Output	Eigenvalues

No.	Argument	Type	Size	Input/ Output	Contents
8	M	I	1	Input	Maximum number of the eigenvalues to be computed
				Output	Number of the obtained eigenvalues
9	E1	$\left\{\begin{matrix}\text{D}\\\text{R}\end{matrix}\right\}$	1	Input	E1<E2: Obtain M eigenvalues in the interval [E1, E2] from the smallest one. (E2 is upper bound.) E1>E2: Obtain M eigenvalues in the interval [E1, E2] from the largest one. (E2 is lower bound.) (See Notes (c) and (d))
10	E2	$\left\{\begin{matrix}\text{D}\\\text{R}\end{matrix}\right\}$	1	Input	
11	VE	$\left\{\begin{matrix}\text{D}\\\text{R}\end{matrix}\right\}$	LNV, M	Output	Eigenvectors (column vector) corresponding to each eigenvalue
12	LNV	I	1	Input	Adjustable dimension of array VE
13	IW1	I	M	Output	Eigenvector flag (See Note (a))
14	W1	$\left\{\begin{matrix}\text{D}\\\text{R}\end{matrix}\right\}$	9 × N	Work	Work area
15	IERR	I	1	Output	Error indicator

(4) **Restrictions**

(a) $0 < \text{N} \leq \text{LNA}, \text{LNB}, \text{LNV}$

(b) $0 < \text{M} \leq \text{N}$

(5) **Error indicator**

IERR value	Meaning	Processing
0	Normal termination.	
1000	N was equal to 1.	$\text{E}(1) \leftarrow \text{A}(1,1)/\text{B}(1,1)$ and $\text{VE}(1,1) \leftarrow 1.0/\sqrt{\text{B}(1,1)}$ are performed.
1500	The number of eigenvalues between E1 and E2 is less than M.	All the eigenvalues and the corresponding eigenvectors between E1 and E2 are obtained and the number of the found eigenvalue is output to M.
2000	The maximum number of iterations was exceeded by the inverse iterations for obtaining eigenvectors.	Some eigenvectors are obtained with low precision, and processing continues. (See Note (e).)
2100	*B* has a diagonal element very close to zero.	Some eigenvectors may be obtained with low precision, and processing continues.
3000	Restriction (a) or (b) was not satisfied.	Processing is aborted.
4000	*B* was not positive definite.	

(6) **Notes**

(a) Data should be stored only in the upper triangular portions of arrays A and B.

(b) If EPS ≤ 0, the optimum value is automatically set internally. Normally, a negative value should be set so that this value will be set automatically. EPS is used to obtain eigenvalues by using the Bisection method.

(c) If E1 $<$ E2 the obtained eigenvalues and eigenvectors are stored in ascending order. On the other hand, if E1>E2 the eigenvalues and eigenvectors are stored in descending order.

(d) If E1 $=$ E2, the eigenvalues in the interval [E1 $-$ EPS, E1 $+$ EPS] are obtained. Normally, E1 should be set to be different E2.

(e) If the maximum number of iterations is exceeded when using the inverse iteration method (IERR $=$ 2000 is output), the following processing is performed.

If IW1(i) $=$ 0: The i-th eigenvector calculation is normally terminated.

If IW1(i) $\neq$ 0: The convergence condition is not satisfied for the i-th eigenvector calculation, and the eigenvector precision is low. In this case, the iteration count is set for IW1(i).

If processing is normally terminated (IERR $=$ 0 is output), IW1(i) $=$ 0 is set.

(f) Eigenvectors $\boldsymbol{v}_i$ are an orthonormal set so that $\boldsymbol{v}_j^T B\boldsymbol{v}_k = \delta_{j,k}$

(g) If eigenvectors are not required, use 4.14.6 $\left\{\begin{matrix}\text{DCGSEN}\\ \text{RCGSEN}\end{matrix}\right\}$.

(7) **Example**

(a) ProblemObtain the two eigenvalues in the interval [0.001, 0.1] from the smallest one of $A\boldsymbol{x} = \lambda B\boldsymbol{x}$, where matrices A and B are as follows:

$$A = \begin{bmatrix} 611 & 196 & -192 & 407 & -8 & -52 & -49 & 29 \\ 196 & 899 & 113 & -192 & -71 & -43 & -8 & -44 \\ -192 & 113 & 899 & 196 & 61 & 49 & 8 & 52 \\ 407 & -192 & 196 & 611 & 8 & 44 & 59 & -23 \\ -8 & -71 & 61 & 8 & 411 & -599 & 208 & 208 \\ -52 & -43 & 49 & 44 & -599 & 411 & 208 & 208 \\ -49 & -8 & 8 & 59 & 208 & 208 & 99 & -911 \\ 29 & -44 & 52 & -23 & 208 & 208 & -911 & 99 \end{bmatrix}$$

$$B = \begin{bmatrix} 170 & 18 & 33 & -21 & -17 & 13 & 25 & -36 \\ 18 & 171 & -21 & 22 & 13 & -17 & -36 & 25 \\ 33 & -21 & 171 & 18 & 25 & -36 & -17 & 13 \\ -21 & 22 & 18 & 171 & -36 & 25 & 13 & -17 \\ -17 & 13 & 25 & -36 & 171 & 18 & 33 & -21 \\ 13 & -17 & -36 & 25 & 18 & 171 & -21 & -3 \\ 25 & -36 & -17 & 13 & 33 & -21 & 171 & 18 \\ -36 & 25 & 13 & -17 & -21 & -3 & 18 & 171 \end{bmatrix}$$

and their corresponding eigenvectors.

(b) Input data

Matrix A, LNA=11, N=8, matrix B, LNB=11, EPS=−1.0, M=2, E1=0.001, E2=0.1 and LNV=10.

(c) Main program

```
      PROGRAM BCGSEE
! *** EXAMPLE OF DCGSEE ***
      IMPLICIT REAL(8) (A-H,O-Z)
      PARAMETER ( LNA = 11, LNB = 11, LNV = 10 )
      DIMENSION A(LNA,LNA), B(LNB,LNB), E(LNA), VE(LNV,LNV),&
                 IW1(LNA), W1(9*LNA)
!
      READ(5,*) N, M, E1, E2
      DO 10 J=1, N
         READ(5,*) (A(J,I), I=J, N)
   10 CONTINUE
      DO 20 J=1, N
         READ(5,*) (B(J,I), I=J, N)
   20 CONTINUE
!
      WRITE(6,1000) N, M, E1, E2
      WRITE(6,1100) 'A'
      DO 30 J=1, N
         WRITE(6,1200) (A(I,J), I=1, J-1), (A(J,I), I=J, N)
   30 CONTINUE
      WRITE(6,1100) 'B'
      DO 40 J=1, N
         WRITE(6,1200) (B(I,J), I=1, J-1), (B(J,I), I=J, N)
   40 CONTINUE
!
      EPS = -1.0D0
!
      CALL DCGSEE(A,LNA,N,B,LNB,EPS,E,M,E1,E2,VE,LNV,IW1,W1,IERR)
!
      WRITE(6,1300) IERR
!
      DO 60 K=1, M-3, 4
         WRITE(6,1400) ('EIGENVALUE ', I=1, 4)
         WRITE(6,1500) (E(I), I=K, K+3)
         WRITE(6,1400) ('EIGENVECTOR', I=1, 4)
         DO 50 J=1, N
            WRITE(6,1500) (VE(J,I), I=K, K+3)
   50    CONTINUE
   60 CONTINUE
      IF(MOD(M,4).NE.0) THEN
         WRITE(6,1400) ('EIGENVALUE ', I=M/4*4+1, M)
         WRITE(6,1500) (E(I), I=M/4*4+1, M)
         WRITE(6,1400) ('EIGENVECTOR', I=M/4*4+1, M)
         DO 70 J=1, N
            WRITE(6,1500) (VE(J,I), I=M/4*4+1, M)
   70    CONTINUE
      ENDIF
      STOP
!
 1000 FORMAT(1X,/,/,&
             1X,'***  DCGSEE  ***',/,/,&
             1X,' **  INPUT  **',/,/,&
             1X,'N = ', I4,'  M = ', I4,/,/,&
             1X,'E1= ', 1PD14.7,'  E2= ', 1PD14.7)
 1100 FORMAT(1X,/,&
             1X,'     INPUT MATRIX ',A1,/)
 1200 FORMAT(1X, 6X, 11(F8.1))
 1300 FORMAT(1X,/,/,&
             1X,' **  OUTPUT  **',/,/,&
             1X,'     IERR = ', I4)
 1400 FORMAT(1X,/,1X, 4(5X, A11, 2X))
 1500 FORMAT(1X, 4(2X, 1PD14.7, 2X))
      END
```

(d) Output results

```
***  DCGSEE  ***

 **  INPUT  **

N =    4  M =    2

E1=  1.0000000D-03  E2=  1.0000000D-01

     INPUT MATRIX A

           2.0     1.0     1.0     2.0
           1.0     1.0     1.0     1.0
           1.0     1.0     2.0     2.0
           2.0     1.0     2.0     4.0

     INPUT MATRIX B

         153.0    31.0    58.0   -58.0
          31.0   153.0   -58.0    58.0
          58.0   -58.0   153.0    31.0
         -58.0    58.0    31.0   153.0
```

```
**  OUTPUT  **

    IERR =    0

    EIGENVALUE        EIGENVALUE
  5.3688291D-03     2.7447086D-02

    EIGENVECTOR       EIGENVECTOR
 -4.9839235D-02     1.6115522D-02
 -3.7709049D-02    -6.8634504D-02
  1.9357438D-02    -8.5979167D-02
  3.3245027D-02    -1.4513123D-03
```

4.14.6 DCGSEN, RCGSEN Eigenvalues in an Interval of a Real Symmetric Matrix (Interval Specified) (Generalized Eigenvalue Problem $Ax = \lambda Bx$, B: Positive)

(1) **Function**

DCGSEN or RCGSEN uses the Cholesky method to transform the real symmetric matrix (two-dimensional array type) (upper triangular type) generalized eigenvalue problem $A\boldsymbol{x} = \lambda B\boldsymbol{x}$ (A: Real symmetric matrix, B: Positive symmetric matrix) to a standard eigenvalue problem and uses the Householder method and the Bisection method to obtain the m largest eigenvalues or m smallest eigenvalues in a specified interval.

(2) **Usage**

Double precision:
  CALL DCGSEN (A, LNA, N, B, LNB, EPS, E, M, E1, E2, W1, IERR)
Single precision:
  CALL RCGSEN (A, LNA, N, B, LNB, EPS, E, M, E1, E2, W1, IERR)

(3) **Arguments**

D:Double precision real Z:Double precision complex I: $\left\{\begin{array}{l}\text{INTEGER(4) as for 32bit Integer}\\\text{INTEGER(8) as for 64bit Integer}\end{array}\right\}$
R:Single precision real C:Single precision complex

No.	Argument	Type	Size	Input/ Output	Contents
1	A	$\left\{\begin{array}{c}D\\R\end{array}\right\}$	LNA, N	Input	Real symmetric matrix A (two-dimensional array type) (upper triangular type)
				Output	Input-time contents are not retained.
2	LNA	I	1	Input	Adjustable dimension of array A
3	N	I	1	Input	Order of matrix A and B
4	B	$\left\{\begin{array}{c}D\\R\end{array}\right\}$	LNB, N	Input	Positive symmetric matrix B (two-dimensional array type) (upper triangular type)
				Output	Input-time contents are not retained.
5	LNB	I	1	Input	Adjustable dimension of array B
6	EPS	$\left\{\begin{array}{c}D\\R\end{array}\right\}$	1	Input	Parameter that assigns an upper limit to the absolute error for use in the eigenvalue convergence test. (See Note (b))
7	E	$\left\{\begin{array}{c}D\\R\end{array}\right\}$	M	Output	Eigenvalues
8	M	I	1	Input	Maximum number of the eigenvalues to be computed
				Output	Number of the obtained eigenvalues

No.	Argument	Type	Size	Input/ Output	Contents
9	E1	$\left\{\begin{matrix}\mathrm{D}\\\mathrm{R}\end{matrix}\right\}$	1	Input	E1<E2: Obtain M eigenvalues in the interval [E1, E2] from the smallest one. (E2 is upper bound.) E1>E2: Obtain M eigenvalues in the interval [E1, E2] from the largest one. (E2 is lower bound.) (See Notes (c) and (d))
10	E2	$\left\{\begin{matrix}\mathrm{D}\\\mathrm{R}\end{matrix}\right\}$	1	Input	
11	W1	$\left\{\begin{matrix}\mathrm{D}\\\mathrm{R}\end{matrix}\right\}$	5 × N	Work	Work area
12	IERR	I	1	Output	Error indicator

(4) **Restrictions**

(a) $0 < \mathrm{N} \leq \mathrm{LNA}, \mathrm{LNB}$

(b) $0 < \mathrm{M} \leq \mathrm{N}$

(5) **Error indicator**

IERR value	Meaning	Processing
0	Normal termination.	
1000	N was equal to 1.	$\mathrm{E}(1) \leftarrow \mathrm{A}(1,1)/\mathrm{B}(1,1)$ is performed.
1500	The number of eigenvalues between E1 and E2 is less than M.	All the eigenvalues and the corresponding eigenvectors between E1 and E2 are obtained and the number of the found eigenvalue is output to M.
2100	B has a diagonal element very close to zero.	Processing continues.
3000	Restriction (a) or (b) was not satisfied.	Processing is aborted.
4000	B was not positive definite.	

(6) **Notes**

(a) Data should be stored only in the upper triangular portions of arrays A and B.

(b) If EPS ≤ 0, the optimum value is automatically set internally. Normally, a negative value should be set so that this value will be set automatically. EPS is used to obtain eigenvalues by using the Bisection method.

(c) If E1 < E2 the obtained eigenvalues and eigenvectors are stored in ascending order. On the other hand, if E1 > E2 the eigenvalues and eigenvectors are stored in descending order.

(d) If E1 = E2, the eigenvalues in the interval [E1 − EPS, E1 + EPS] are obtained. Normally, E1 should be set to be different E2.

4.15 GENERALIZED EIGENVALUE PROBLEM ($ABx = \lambda x$) FOR REAL SYMMETRIC MATRICES (TWO-DIMENSIONAL ARRAY TYPE) (UPPER TRIANGULAR TYPE)

4.15.1 DCGJAA, RCGJAA All Eigenvalues and All Eigenvectors of Real Symmetric Matrices (Generalized Eigenvalue Problem $ABx = \lambda x$, B: Positive)

(1) **Function**

Generalized eigenvalue problem

$$AB\boldsymbol{x} = \lambda\boldsymbol{x}$$

(A: Real symmetric, B: Positive real symmetric) is solved by using the Cholesky method, the Householder method and QR method to obtain all eigenvalues λ and corresponding all eigenvectors $\boldsymbol{x}$.

(2) **Usage**

Double precision:
CALL DCGJAA (A, LNA, N, B, LNB, E, WORK, IERR)
Single precision:
CALL RCGJAA (A, LNA, N, B, LNB, E, WORK, IERR)

(3) **Arguments**

D:Double precision real Z:Double precision complex I: $\left\{\begin{array}{l}\text{INTEGER(4) as for 32bit Integer}\\ \text{INTEGER(8) as for 64bit Integer}\end{array}\right\}$
R:Single precision real C:Single precision complex

No.	Argument	Type	Size	Input/ Output	Contents
1	A	$\left\{\begin{matrix}\text{D}\\ \text{R}\end{matrix}\right\}$	LNA, N	Input	Real symmetric matrix A
				Output	Eigenvectors $\boldsymbol{x}$
2	LNA	I	1	Input	Adjustable dimension of array A
3	N	I	1	Input	Order of matrices A and B
4	B	$\left\{\begin{matrix}\text{D}\\ \text{R}\end{matrix}\right\}$	LNB, N	Input	Real symmetric matrix B
				Output	Input-time contents are not retained.
5	LNB	I	1	Input	Adjustable dimension of array B
6	E	$\left\{\begin{matrix}\text{D}\\ \text{R}\end{matrix}\right\}$	N	Output	Eigenvalues λ
7	WORK	$\left\{\begin{matrix}\text{D}\\ \text{R}\end{matrix}\right\}$	2 × N	Work	Work area
8	IERR	I	1	Output	Error indicator

(4) **Restrictions**

(a) $1 \leq \text{N} \leq \text{LNA}, \text{LNB}$

(5) **Error indicator**

IERR value	Meaning	Processing
0	Normal termination.	
1000	N was equal to 1.	$\mathrm{E}(1) \leftarrow \mathrm{A}(1,1) \times \mathrm{B}(1,1)$ and $\mathrm{A}(1,1) \leftarrow 1.0/\sqrt{\mathrm{B}(1,1)}$ are performed.
2100	B has a diagonal element very close to zero.	Some results may be obtained with low precision.
3000	Restriction (a) was not satisfied.	Processing is aborted.
4000	B was not positive definite.	
$5000 + i$	The sequence did not converge in the step where the eigenvalue was obtained. $(1 \le i \le \mathrm{N})$	Eigenvalues obtained by this time are entered in $\mathrm{E}(1), \cdots, \mathrm{E}(\mathrm{i}-1)$ (However, the order is irregular). No eigenvector is obtained at this time.

(6) **Notes**

(a) Arrays A and B should be stored only in the upper triangular portions.

(b) Eigenvalues are stored in ascending order.

(c) Eigenvectors $\boldsymbol{v}_i$ are an orthonormal set so that $\boldsymbol{v}_j^T B \boldsymbol{v}_k = \delta_{j,k}$

(d) 4.15.2 $\left\{\begin{matrix}\text{DCGJAN}\\ \text{RCGJAN}\end{matrix}\right\}$ should be used if the eigenvectors are not needed.

(e) 4.16.1 $\left\{\begin{matrix}\text{DCGKAA}\\ \text{RCGKAA}\end{matrix}\right\}$ should be used if matrix A is only positive.

(7) **Example**

(a) Problem

Obtain all eigenvalues and their corresponding eigenvectors non-symmetric matrix AB when A and B are positive symmetric matrices.

$$A = \begin{bmatrix} 1.07692 & 0.28571 & 0.09733 & 0.04887 \\ 0.28571 & 1.02041 & 0.26316 & 0.08610 \\ 0.09733 & 0.26316 & 1.00917 & 0.25676 \\ 0.04887 & 0.08610 & 0.25676 & 1.00518 \end{bmatrix}$$

$$B = \begin{bmatrix} 1.04762 & 0.18841 & 0.05996 & 0.02968 \\ 0.18841 & 1.01235 & 0.17460 & 0.05314 \\ 0.05996 & 0.17460 & 1.00552 & 0.17073 \\ 0.02968 & 0.05314 & 0.17073 & 1.00312 \end{bmatrix}$$

Note Where matrix elements of A and B are defined as

$$A = P(3.0, 4), \quad B = P(5.0, 4)$$

where

$$P(a^2, \mathrm{N})_{i,j} = 2\int_0^\infty \cos(ait)\cos(ajt)e^{-t}dt (1 \le i \le \mathrm{N}; 1 \le j \le \mathrm{N}).$$

All eigenvalues of the each matrix exist within a finite interval which is independent of N.

(b) Input data

N=4,LNA=LNB=4 and Symmetric matrices A and B.

(c) Main program

```
      PROGRAM BCGJAA
! *** EXAMPLE OF DCGJAA ***
      IMPLICIT NONE
!
      INTEGER N,LNA,LNB
      PARAMETER( N = 4, LNA = 4, LNB = 4 )
      INTEGER IERR,I,J,L
      REAL(8) A(LNA,N),B(LNB,N)
      REAL(8) E(N),WORK(2*N)
      REAL(8) ONE,TRE,FIV
      PARAMETER( ONE = 1.D0, TRE = 3.D0, FIV = 5.D0 )
!
      WRITE(6,6000) N, LNA, LNB
      DO 100 I=1,N
      DO 110 J=1,N
         A(I,J)= ONE/(ONE+TRE*DBLE(I+J)**2)+ONE/(ONE+TRE*DBLE(I-J)**2)
         B(I,J)= ONE/(ONE+FIV*DBLE(I+J)**2)+ONE/(ONE+FIV*DBLE(I-J)**2)
  110 CONTINUE
  100 CONTINUE
      WRITE(6,6010)
      DO 120 I=1,N
         WRITE(6,6020) A(I,1),A(I,2),A(I,3),A(I,4)
  120 CONTINUE
      WRITE(6,6030)
      DO 130 I=1,N
         WRITE(6,6020) B(I,1),B(I,2),B(I,3),B(I,4)
  130 CONTINUE
!
      CALL DCGJAA(A, LNA, N, B, LNB, E, WORK, IERR)
!
      WRITE(6,6040) IERR
      DO 140 I=1,N,2
         WRITE(6,6050) (' EIGENVALUE',L=1,2)
         WRITE(6,6060) E(I),E(I+1)
         WRITE(6,6050) ('EIGENVECTOR',L=1,2)
         WRITE(6,6060) A(1,I),A(1,I+1)
         WRITE(6,6060) A(2,I),A(2,I+1)
         WRITE(6,6060) A(3,I),A(3,I+1)
         WRITE(6,6060) A(4,I),A(4,I+1)
  140 CONTINUE
!
      STOP
 6000 FORMAT(/,&
             1X,'***  DCGJAA  ***',/,/,&
             1X,' **  INPUT  **',/,/,&
             1X,'     N = ',I4, '    LNA = ',I4,'  LNB = ',I4,/)
 6010 FORMAT(/,&
             1X,'     INPUT MATRIX A',/)
 6020 FORMAT(1X,3X,4(2X,F9.5))
 6030 FORMAT(/,&
             1X,'     INPUT MATRIX B',/)
 6040 FORMAT(/,&
             1X,' **  OUTPUT  **',/,/,&
             1X,'     IERR =',I5,/)
 6050 FORMAT(/,&
             1X,7X,A11,22X,A11)
 6060 FORMAT(1X,5X,1PD14.7,19X,1PD14.7)
      END
```

(d) Output results

```
***  DCGJAA  ***

 **  INPUT  **

     N =    4    LNA =    4  LNB =    4


     INPUT MATRIX A

       1.07692    0.28571    0.09733    0.04887
       0.28571    1.02041    0.26316    0.08610
       0.09733    0.26316    1.00917    0.25676
       0.04887    0.08610    0.25676    1.00518

     INPUT MATRIX B

       1.04762    0.18841    0.05996    0.02968
       0.18841    1.01235    0.17460    0.05314
       0.05996    0.17460    1.00552    0.17073
       0.02968    0.05314    0.17073    1.00312

 **  OUTPUT  **

     IERR =    0
```

```
   EIGENVALUE                       EIGENVALUE
 5.0334859D-01                    7.0649951D-01

  EIGENVECTOR                      EIGENVECTOR
-3.5988845D-01                   -5.7276742D-01
 7.0243936D-01                    4.9740733D-01
-7.1756352D-01                    4.1982158D-01
 4.0575850D-01                   -6.2631989D-01

   EIGENVALUE                       EIGENVALUE
 1.1519432D+00                    2.1334368D+00

  EIGENVECTOR                      EIGENVECTOR
 5.9964094D-01                    4.1404961D-01
 2.5727788D-01                    4.9417207D-01
-3.8926246D-01                    4.5974369D-01
-6.0442734D-01                    3.2426391D-01
```

4.15.2 DCGJAN, RCGJAN All Eigenvalues of Real Symmetric Matrices (Generalized Eigenvalue Problem $ABx = \lambda x$, B: Positive)

(1) Function

Generalized eigenvalue problem

$$AB\boldsymbol{x} = \lambda \boldsymbol{x}$$

(A: Real symmetric, B: Positive real symmetric) is solved by using the Cholesky method, the Householder method and QR method to obtain all eigenvalues λ .

(2) Usage

Double precision:
CALL DCGJAN (A, LNA, N, B, LNB, E, WORK, IERR)
Single precision:
CALL RCGJAN (A, LNA, N, B, LNB, E, WORK, IERR)

(3) Arguments

D:Double precision real Z:Double precision complex
R:Single precision real C:Single precision complex
I: $\left\{\begin{matrix}\text{INTEGER(4) as for 32bit Integer}\\ \text{INTEGER(8) as for 64bit Integer}\end{matrix}\right\}$

No.	Argument	Type	Size	Input/ Output	Contents
1	A	$\left\{\begin{matrix}D\\R\end{matrix}\right\}$	LNA, N	Input	Real symmetric matrix A
				Output	Input-time contents are not retained.
2	LNA	I	1	Input	Adjustable dimension of array A
3	N	I	1	Input	Order of matrices A and B
4	B	$\left\{\begin{matrix}D\\R\end{matrix}\right\}$	LNB, N	Input	Real symmetric matrix B
				Output	Input-time contents are not retained.
5	LNB	I	1	Input	Adjustable dimension of array B
6	E	$\left\{\begin{matrix}D\\R\end{matrix}\right\}$	N	Output	Eigenvalues λ
7	WORK	$\left\{\begin{matrix}D\\R\end{matrix}\right\}$	2 × N	Work	Work area
8	IERR	I	1	Output	Error indicator

(4) Restrictions

(a) $1 \leq \mathrm{N} \leq \mathrm{LNA}, \mathrm{LNB}$

(5) **Error indicator**

IERR value	Meaning	Processing
0	Normal termination.	
1000	N was equal to 1.	$\mathrm{E}(1) \leftarrow \mathrm{A}(1,1) \times \mathrm{B}(1,1)$ is performed.
2100	B has a diagonal element very close to zero.	Some results may be obtained with low precision.
3000	Restriction (a) was not satisfied.	Processing is aborted.
4000	B was not positive definite.	
$5000 + i$	The sequence did not converge in the step where the eigenvalue was obtained. $(1 \leq i \leq \mathrm{N})$	Eigenvalues obtained by this time are entered in $\mathrm{E}(1), \cdots, \mathrm{E}(\mathrm{i}-1)$ (However, the order is irregular).

(6) **Notes**

(a) Arrays A and B should be stored only in the upper triangular portions.

(b) Eigenvalues are stored in ascending order.

(c) 4.15.1 $\left\{\begin{matrix}\text{DCGJAA}\\ \text{RCGJAA}\end{matrix}\right\}$ should be used if the eigenvectors are needed.

(d) 4.16.2 $\left\{\begin{matrix}\text{DCGKAN}\\ \text{RCGKAN}\end{matrix}\right\}$ should be used if matrix A is only positive.

4.16 GENERALIZED EIGENVALUE PROBLEM ($BAx = \lambda x$) FOR REAL SYMMETRIC MATRICES (TWO-DIMENSIONAL ARRAY TYPE) (UPPER TRIANGULAR TYPE)

4.16.1 DCGKAA, RCGKAA All Eigenvalues and All Eigenvectors of Real Symmetric Matrices (Generalized Eigenvalue Problem $BAx = \lambda x$, B: Positive)

(1) **Function**

Generalized eigenvalue problem

$$BA\boldsymbol{x} = \lambda \boldsymbol{x}$$

(A: Real symmetric, B: Positive real symmetric) is solved by using the Cholesky method, the Householder method and QR method to obtain all eigenvalues λ and corresponding all eigenvectors $\boldsymbol{x}$.

(2) **Usage**

Double precision:
CALL DCGKAA (A, LNA, N, B, LNB, E, WORK, IERR)
Single precision:
CALL RCGKAA (A, LNA, N, B, LNB, E, WORK, IERR)

(3) **Arguments**

D:Double precision real	Z:Double precision complex	I: $\left\{\begin{array}{l}\text{INTEGER(4) as for 32bit Integer}\\ \text{INTEGER(8) as for 64bit Integer}\end{array}\right\}$
R:Single precision real	C:Single precision complex	

No.	Argument	Type	Size	Input/ Output	Contents
1	A	$\left\{\begin{matrix}\text{D}\\ \text{R}\end{matrix}\right\}$	LNA, N	Input	Real symmetric matrix A
				Output	Eigenvectors $\boldsymbol{x}$
2	LNA	I	1	Input	Adjustable dimension of array A
3	N	I	1	Input	Order of matrices A and B
4	B	$\left\{\begin{matrix}\text{D}\\ \text{R}\end{matrix}\right\}$	LNB, N	Input	Real symmetric matrix B
				Output	Input-time contents are not retained.
5	LNB	I	1	Input	Adjustable dimension of array B
6	E	$\left\{\begin{matrix}\text{D}\\ \text{R}\end{matrix}\right\}$	N	Output	Eigenvalues λ
7	WORK	$\left\{\begin{matrix}\text{D}\\ \text{R}\end{matrix}\right\}$	2 × N	Work	Work area
8	IERR	I	1	Output	Error indicator

(4) **Restrictions**

(a) $1 \le \mathrm{N} \le \mathrm{LNA}, \mathrm{LNB}$

(5) **Error indicator**

IERR value	Meaning	Processing
0	Normal termination.	
1000	N was equal to 1.	$\mathrm{E}(1) \leftarrow \mathrm{A}(1,1) \times \mathrm{B}(1,1)$ and $\mathrm{A}(1,1) \leftarrow \sqrt{\mathrm{B}(1,1)}$ are performed.
2100	B has a diagonal element very close to zero.	Some results may be obtained with low precision.
3000	Restriction (a) was not satisfied.	Processing is aborted.
4000	B was not positive definite.	
$5000 + i$	The sequence did not converge in the step where the eigenvalue was obtained. $(1 \le i \le \mathrm{N})$	Eigenvalues obtained by this time are entered in $\mathrm{E}(1), \cdots, \mathrm{E}(\mathrm{i}-1)$ (However, the order is irregular). No eigenvector is obtained at this time.

(6) **Notes**

(a) Arrays A and B should be stored only in the upper triangular portions.

(b) Eigenvalues are stored in ascending order.

(c) Eigenvectors $\boldsymbol{v}_i$ are an orthonormal set so that $\boldsymbol{v}_j^T B^{-1} \boldsymbol{v}_k = \delta_{j,k}$

(d) 4.16.2 $\begin{Bmatrix} \text{DCGKAN} \\ \text{RCGKAN} \end{Bmatrix}$ should be used if the eigenvectors are not needed.

(e) 4.15.1 $\begin{Bmatrix} \text{DCGJAA} \\ \text{RCGJAA} \end{Bmatrix}$ should be used if matrix A is only positive.

(7) **Example**

(a) Problem

Obtain all eigenvalues and their corresponding eigenvectors non-symmetric matrix AB when A and B are positive symmetric matrices.

$$A = \begin{bmatrix} 1.07692 & 0.28571 & 0.09733 & 0.04887 \\ 0.28571 & 1.02041 & 0.26316 & 0.08610 \\ 0.09733 & 0.26316 & 1.00917 & 0.25676 \\ 0.04887 & 0.08610 & 0.25676 & 1.00518 \end{bmatrix}$$

$$B = \begin{bmatrix} 1.04762 & 0.18841 & 0.05996 & 0.02968 \\ 0.18841 & 1.01235 & 0.17460 & 0.05314 \\ 0.05996 & 0.17460 & 1.00552 & 0.17073 \\ 0.02968 & 0.05314 & 0.17073 & 1.00312 \end{bmatrix}$$

Note Where matrix elements of A and B are defined as

$$A = P(3.0, 4), \quad B = P(5.0, 4)$$

where

$$P(a^2, \mathrm{N})_{i,j} = 2\int_0^\infty \cos(ait)\cos(ajt)e^{-t}dt(1 \le i \le \mathrm{N}; 1 \le j \le \mathrm{N}).$$

All eigenvalues of the each matrix exist within a finite interval which is independent of N.

(b) Input data

N=4,LNA=LNB=4 and Symmetric matrices A and B.

(c) Main program

```
      PROGRAM BCGKAA
! *** EXAMPLE OF DCGKAA ***
      IMPLICIT NONE
!
      INTEGER N,LNA,LNB
      PARAMETER( N = 4, LNA = 4, LNB = 4 )
      INTEGER IERR,I,J,L
      REAL(8) A(LNA,N),B(LNB,N)
      REAL(8) E(N),WORK(2*N)
      REAL(8) ONE, TRE, FIV
      PARAMETER( ONE = 1.D0, TRE = 3.D0, FIV = 5.D0 )
!
      WRITE(6,6000) N, LNA, LNB
      DO 100 I=1,N
      DO 110 J=1,N
         A(I,J)= ONE/(ONE+TRE*DBLE(I+J)**2)+ONE/(ONE+TRE*DBLE(I-J)**2)
         B(I,J)= ONE/(ONE+FIV*DBLE(I+J)**2)+ONE/(ONE+FIV*DBLE(I-J)**2)
  110 CONTINUE
  100 CONTINUE
      WRITE(6,6010)
      DO 120 I=1,N
         WRITE(6,6020) A(I,1),A(I,2),A(I,3),A(I,4)
  120 CONTINUE
      WRITE(6,6030)
      DO 130 I=1,N
         WRITE(6,6020) B(I,1),B(I,2),B(I,3),B(I,4)
  130 CONTINUE
!
      CALL DCGKAA(A, LNA, N, B, LNB, E, WORK, IERR)
!
      WRITE(6,6040) IERR
      DO 140 I=1,N,2
         WRITE(6,6050) (' EIGENVALUE',L=1,2)
         WRITE(6,6060) E(I),E(I+1)
         WRITE(6,6050) ('EIGENVECTOR',L=1,2)
         WRITE(6,6060) A(1,I),A(1,I+1)
         WRITE(6,6060) A(2,I),A(2,I+1)
         WRITE(6,6060) A(3,I),A(3,I+1)
         WRITE(6,6060) A(4,I),A(4,I+1)
  140 CONTINUE
!
      STOP
 6000 FORMAT(/,&
             1X,'***  DCGKAA  ***',/,/,&
             1X,' **  INPUT  **',/,/,&
             1X,'     N = ',I4, '    LNA = ',I4,'  LNB = ',I4,/)
 6010 FORMAT(/,&
             1X,'     INPUT MATRIX A',/)
 6020 FORMAT(1X,3X,4(2X,F9.5))
 6030 FORMAT(/,&
             1X,'     INPUT MATRIX B',/)
 6040 FORMAT(/,&
             1X,' **  OUTPUT  **',/,/,&
             1X,'     IERR =',I5,/)
 6050 FORMAT(/,&
             1X,7X,A11,22X,A11)
 6060 FORMAT(1X,5X,1PD14.7,19X,1PD14.7)
      END
```

(d) Output results

```
***  DCGKAA  ***

 **  INPUT  **

     N =    4    LNA =    4  LNB =    4


     INPUT MATRIX A

       1.07692    0.28571    0.09733    0.04887
       0.28571    1.02041    0.26316    0.08610
       0.09733    0.26316    1.00917    0.25676
       0.04887    0.08610    0.25676    1.00518

     INPUT MATRIX B

       1.04762    0.18841    0.05996    0.02968
```

```
        0.18841    1.01235    0.17460    0.05314
        0.05996    0.17460    1.00552    0.17073
        0.02968    0.05314    0.17073    1.00312

 **  OUTPUT  **

     IERR =    0


         EIGENVALUE                        EIGENVALUE
      5.0334859D-01                     7.0649951D-01

        EIGENVECTOR                       EIGENVECTOR
     -2.7566971D-01                    -4.9973959D-01
      5.3958111D-01                     4.3565256D-01
     -5.5118459D-01                     3.6771142D-01
      3.1116215D-01                    -5.4715726D-01

         EIGENVALUE                        EIGENVALUE
      1.1519432D+00                     2.1334368D+00

        EIGENVECTOR                       EIGENVECTOR
      6.3538913D-01                     5.6406228D-01
      2.7334189D-01                     6.7578767D-01
     -4.1372916D-01                     6.2875822D-01
     -6.4130226D-01                     4.4231632D-01
```

4.16.2 DCGKAN, RCGKAN All Eigenvalues of Real Symmetric Matrices (Generalized Eigenvalue Problem $BAx = \lambda x$, B: Positive)

(1) **Function**

Generalized eigenvalue problem

$$BA\boldsymbol{x} = \lambda \boldsymbol{x}$$

(A: Real symmetric, B: Positive real symmetric) is solved by using the Cholesky method, the Householder method and QR method to obtain all eigenvalues λ .

(2) **Usage**

Double precision:
  CALL DCGKAN (A, LNA, N, B, LNB, E, WORK, IERR)
Single precision:
  CALL RCGKAN (A, LNA, N, B, LNB, E, WORK, IERR)

(3) **Arguments**

D:Double precision real Z:Double precision complex I: $\left\{\begin{array}{l}\text{INTEGER(4) as for 32bit Integer}\\ \text{INTEGER(8) as for 64bit Integer}\end{array}\right\}$
R:Single precision real C:Single precision complex

No.	Argument	Type	Size	Input/ Output	Contents
1	A	$\left\{\begin{matrix}\text{D}\\ \text{R}\end{matrix}\right\}$	LNA, N	Input	Real symmetric matrix A
				Output	Input-time contents are not retained.
2	LNA	I	1	Input	Adjustable dimension of array A
3	N	I	1	Input	Order of matrices A and B
4	B	$\left\{\begin{matrix}\text{D}\\ \text{R}\end{matrix}\right\}$	LNB, N	Input	Real symmetric matrix B
				Output	Input-time contents are not retained.
5	LNB	I	1	Input	Adjustable dimension of array B
6	E	$\left\{\begin{matrix}\text{D}\\ \text{R}\end{matrix}\right\}$	N	Output	Eigenvalues λ
7	WORK	$\left\{\begin{matrix}\text{D}\\ \text{R}\end{matrix}\right\}$	2 × N	Work	Work area
8	IERR	I	1	Output	Error indicator

(4) **Restrictions**

(a) $1 \leq \text{N} \leq \text{LNA}, \text{LNB}$

(5) **Error indicator**

IERR value	Meaning	Processing
0	Normal termination.	
1000	N was equal to 1.	E(1) ← A(1, 1) × B(1, 1) is performed.
2100	B has a diagonal element very close to zero.	Some results may be obtained with low precision.
3000	Restriction (a) was not satisfied.	Processing is aborted.
4000	B was not positive definite.	
$5000 + i$	The sequence did not converge in the step where the eigenvalue was obtained. $(1 \leq i \leq \mathrm{N})$	Eigenvalues obtained by this time are entered in E(1), $\cdots$, E(i − 1) (However, the order is irregular).

(6) **Notes**

(a) Arrays A and B should be stored only in the upper triangular portions.

(b) Eigenvalues are stored in ascending order.

(c) 4.16.1 $\left\{\begin{matrix}\text{DCGKAA}\\\text{RCGKAA}\end{matrix}\right\}$ should be used if the eigenvectors are needed.

(d) 4.15.2 $\left\{\begin{matrix}\text{DCGJAN}\\\text{RCGJAN}\end{matrix}\right\}$ should be used if matrix A is only positive.

4.17 GENERALIZED EIGENVALUE PROBLEM ($Az = \lambda Bz$) FOR HERMITIAN MATRICES (TWO-DIMENSIONAL ARRAY TYPE) (UPPER TRIANGULAR TYPE) (REAL ARGUMENT TYPE)

4.17.1 ZCGRAA, CCGRAA
All Eigenvalues and All Eigenvectors of Hermitian Matrices (Generalized Eigenvalue Problem $Az = \lambda Bz$, B: Positive)

(1) **Function**

Generalized eigenvalue problem

$$A\boldsymbol{z} = \lambda B\boldsymbol{z}$$

(A: Hermitian, B: Positive Hermitian) is solved by using the Cholesky method, the Householder method and QR method to obtain all eigenvalues λ and corresponding all eigenvectors $\boldsymbol{z}$.

(2) **Usage**

Double precision:

CALL ZCGRAA (AR, AI, LNA, N, BR, BI, LNB, E, WORK, IERR)

Single precision:

CALL CCGRAA (AR, AI, LNA, N, BR, BI, LNB, E, WORK, IERR)

(3) **Arguments**

D:Double precision real	Z:Double precision complex	I: $\left\{ \begin{array}{l} \text{INTEGER(4) as for 32bit Integer} \\ \text{INTEGER(8) as for 64bit Integer} \end{array} \right\}$
R:Single precision real	C:Single precision complex	

No.	Argument	Type	Size	Input/ Output	Contents
1	AR	$\left\{\begin{matrix}\text{D}\\\text{R}\end{matrix}\right\}$	LNA, N	Input	Real part of Hermitian matrix A
				Output	Real part of eigenvector z
2	AI	$\left\{\begin{matrix}\text{D}\\\text{R}\end{matrix}\right\}$	LNA, N	Input	Imaginary part of Hermitian matrix A
				Output	Imaginary part of eigenvector z
3	LNA	I	1	Input	Adjustable dimension of arrays AR and AI
4	N	I	1	Input	Order of matrices A and B
5	BR	$\left\{\begin{matrix}\text{D}\\\text{R}\end{matrix}\right\}$	LNB, N	Input	Real part of Hermitian matrix B
				Output	Input-time contents are not retained.

No.	Argument	Type	Size	Input/ Output	Contents
6	BI	$\begin{Bmatrix}\text{D}\\\text{R}\end{Bmatrix}$	LNB, N	Input	Imaginary part of Hermitian matrix B
				Output	Input-time contents are not retained.
7	LNB	I	1	Input	Adjustable dimension of arrays BR and BI
8	E	$\begin{Bmatrix}\text{D}\\\text{R}\end{Bmatrix}$	N	Output	Eigenvalues λ
9	WORK	$\begin{Bmatrix}\text{D}\\\text{R}\end{Bmatrix}$	$4 \times \text{N}$	Work	Work area
10	IERR	I	1	Output	Error indicator

(4) **Restrictions**

(a) $1 \leq \text{N} \leq \text{LNA}, \text{LNB}$

(5) **Error indicator**

IERR value	Meaning	Processing
0	Normal termination.	
1000	N was equal to 1.	$\text{E}(1) \leftarrow \text{AR}(1,1)/\text{BR}(1,1)$, $\text{AR}(1,1) \leftarrow 1.0/\sqrt{\text{BR}(1,1)}$ and $\text{AI}(1,1) \leftarrow 0.0$ are performed.
3000	Restriction (a) was not satisfied.	Processing is aborted.
4000	B was not positive definite.	
5000	The sequence did not converge in the step where the eigenvalue was obtained.	

(6) **Notes**

(a) Arrays AR, AI, BR and BI should be stored only in the upper triangular portions.

(b) Eigenvalues are stored in ascending order.

(c) Eigenvectors $\boldsymbol{v}_i$ are an orthonormal set so that $\boldsymbol{v}_j^* B \boldsymbol{v}_k = \delta_{j,k}$

(d) 4.17.2 $\begin{Bmatrix}\text{ZCGRAN}\\\text{CCGRAN}\end{Bmatrix}$ should be used if the eigenvectors are not needed.

(7) **Example**

(a) Problem

For Hermitian matrix of the degree 4

$$A = \begin{bmatrix} 8 & 3 & 1-2i & -1-2i \\ 3 & 9 & 1+2i & -1+2i \\ 1+2i & 1-2i & 10 & -3 \\ -1+2i & -1-2i & -3 & 11 \end{bmatrix}$$

and its conjugate Hermitian matrix

$$B = \begin{bmatrix} 8 & 3 & 1+2i & -1+2i \\ 3 & 9 & 1-2i & -1-2i \\ 1-2i & 1+2i & 10 & -3 \\ -1-2i & -1+2i & -3 & 11 \end{bmatrix}$$

obtain eigenvector of generalized eigenvalue problem.

(b) Input data

N=4, LNA=4, matrix A, LNB=4 and matrix B.

(c) Main program

```
      PROGRAM ACGRAA
      IMPLICIT REAL(8)(A-H,O-Z)
      PARAMETER ( N=4 , LN=4 )
      REAL(8)      E(N), WORK(4*N)
      COMPLEX(8)  KEEP(LN,N),VM
      REAL(8)      AR(LN,N),BR(LN,N),AI(LN,N),BI(LN,N),SCL(N)
      KEEP(1,1)=( 8.D0, 0.D0)
      KEEP(2,2)=( 9.D0, 0.D0)
      KEEP(3,3)=( 10.D0, 0.D0)
      KEEP(4,4)=( 11.D0, 0.D0)
      KEEP(1,2)=( 3.D0, 0.D0)
      KEEP(1,3)=( 1.D0, 2.D0)
      KEEP(1,4)=(-1.D0, 2.D0)
      KEEP(2,3)=( 1.D0,-2.D0)
      KEEP(2,4)=(-1.D0,-2.D0)
      KEEP(3,4)=(-3.D0, 0.D0)
      METHOD=0
      WRITE(6,60)
      IF(METHOD.EQ.0) WRITE(6,70) N
      DO 1000 I=1,N
      DO 1001 J=I,N
        BR(I,J)=DBLE (KEEP(I,J))
        BI(I,J)=DIMAG(KEEP(I,J))
        KEEP(J,I)=CONJG(KEEP(I,J))
        AR(I,J)=BR(I,J)
        AI(I,J)=-BI(I,J)
        AR(J,I)=AR(I,J)
        AI(J,I)=-AI(I,J)
        BR(J,I)=BR(I,J)
        BI(J,I)=-BI(I,J)
 1001 CONTINUE
 1000 CONTINUE
      IF(METHOD.EQ.0) THEN
         WRITE(6,80)
         DO 1150 I=1,N
         WRITE(6,5000) AR(I,1),AI(I,1),AR(I,2),AI(I,2),&
                       AR(I,3),AI(I,3),AR(I,4),AI(I,4)
 1150    CONTINUE
         WRITE(6,90)
         DO 1250 I=1,N
         WRITE(6,5000) BR(I,1),BI(I,1),BR(I,2),BI(I,2),&
                       BR(I,3),BI(I,3),BR(I,4),BI(I,4)
 1250    CONTINUE
      ENDIF
      IF(METHOD.GT.0) THEN
         CALL BWINTA(AR, AI, BR, BI, LN, N, SCL, IERRA)
         IF(IERRA.GT.0) STOP
         WRITE(6,95)
         DO 1100 I=1,N
         DO 1101 J=I,N
            WRITE(6,6000) AR(I,J),AI(I,J),BR(I,J),BI(I,J),I,J
 1101    CONTINUE
 1100    CONTINUE
      ENDIF
!
      CALL ZCGRAA(AR,AI, LN, N, BR,BI, LN, E, WORK, IERR)
!
```

```
      IF(METHOD.GT.0) THEN
         CALL BWINTB(AR, AI, LN, N, SCL, IERRB)
         IF(IERRB.GT.0) STOP
      ENDIF
      IF(METHOD.GT.0) THEN
         WRITE(6,96)
      ELSE
         WRITE(6,97)
      ENDIF
      WRITE(6,98) IERR
      DO 2000 I=1,N,2
         VM=(0.D0,0.D0)
         DO 2100 J=1,N
         DO 2101 K=1,N
            VM=VM + KEEP(J,K)*CMPLX(AR(K,I), AI(K,I), KIND=8)&
                             *CMPLX(AR(J,I),-AI(J,I), KIND=8)
 2101    CONTINUE
 2100    CONTINUE
         WRITE(6,6500) ('EIGENVALUE ',L=1,2)
         WRITE(6,7000) E(I),E(I+1)
         WRITE(6,6500) ('EIGENVECTOR',L=1,2)
         WRITE(6,8000) AR(1,I),AI(1,I),AR(1,I+1),AI(1,I+1)
         WRITE(6,8000) AR(2,I),AI(2,I),AR(2,I+1),AI(2,I+1)
         WRITE(6,8000) AR(3,I),AI(3,I),AR(3,I+1),AI(3,I+1)
         WRITE(6,8000) AR(4,I),AI(4,I),AR(4,I+1),AI(4,I+1)
 2000 CONTINUE
      STOP
   60 FORMAT(1X,  ' *** ZCGRAA *** ',/,/)
   70 FORMAT(1X, ' *** INPUT ***',/,/,'     N= ',I3,/,/)
   80 FORMAT(1X,'     INPUT MATRIX A ( REAL , IMAGINARY )',/)
   90 FORMAT(1X,/,/,'     INPUT MATRIX B ( REAL , IMAGINARY )',/)
   95 FORMAT(1X,' *** INPUT *** (SCALED) ')
   96 FORMAT(1X,' *** OUTPUT *** (SCALED) ')
   97 FORMAT(1X,/,/, ' *** OUTPUT ***',/,/)
   98 FORMAT(1X,'     IERR = ',I4)
 5000 FORMAT(1X,4('(',F5.1,',',F5.1,')'))
 6000 FORMAT(1X,' RE(A) = ',E10.2,' IM(A) = ',E10.2,&
                ' RE(B) = ',E10.2,' IM(B) = ',E10.2,&
                 ' I,J = ',I2,3X,I2)
 6500 FORMAT(1X ,/, 2(14X,A11,8X))
 7000 FORMAT(1X,2(12X,1PD14.7,7X))
 8000 FORMAT(1X, 2(5X,F12.8,' ,',F12.8,2X))
      END
      SUBROUTINE BWINTA(AR, AI, BR, BI, LN, N, SCL, IERR)
      INTEGER           LN, N
      REAL(8)  AR(LN, N), BR(LN, N), SCL(N),&
                        AI(LN, N), BI(LN, N)
!
      INTEGER I,J
      REAL(8) F,ZERO,ONE
      PARAMETER(ZERO=0.D0)
      PARAMETER(ONE =1.D0)
!
      IERR=0
      IF(N.GT.LN) IERR=3000
      IF(N.LT. 1) IERR=3000
      IF(IERR.GT.0) RETURN
      DO 1000 I=1,N
         IF(BR(I,I).LE.ZERO) THEN
              IERR=4000
              RETURN
         ENDIF
         SCL(I)=ONE/SQRT(BR(I,I))
 1000 CONTINUE
      DO 2000 J=1,N
      DO 2001 I=1,N
         F=SCL(I)*SCL(J)
         AR(I,J)=AR(I,J)*F
         AI(I,J)=AI(I,J)*F
         BR(I,J)=BR(I,J)*F
         BI(I,J)=BI(I,J)*F
 2001 CONTINUE
 2000 CONTINUE
!
      RETURN
      END
!
      SUBROUTINE BWINTB(AR, AI, LN, N, SCL, IERR)
      INTEGER           LN, N
      REAL(8)  AR(LN, N), AI(LN, N), SCL(N)
      INTEGER I,J
!
      IERR=0
      IF(N.GT.LN) IERR=3000
      IF(N.LT. 1) IERR=3000
      IF(IERR.GT.0) RETURN
!
      DO 1000 I=1,N
      DO 1001 J=1,N
         AR(I,J)=AR(I,J)*SCL(I)
         AI(I,J)=AI(I,J)*SCL(I)
 1001 CONTINUE
 1000 CONTINUE
!
      RETURN
      END
```

(d) Output results

```
 *** ZCGRAA ***

 *** INPUT ***

    N=   4

     INPUT MATRIX A ( REAL , IMAGINARY )

(  8.0,  0.0)(  3.0,  0.0)(  1.0, -2.0)( -1.0, -2.0)
(  3.0,  0.0)(  9.0,  0.0)(  1.0,  2.0)( -1.0,  2.0)
(  1.0,  2.0)(  1.0, -2.0)( 10.0,  0.0)( -3.0,  0.0)
( -1.0,  2.0)( -1.0, -2.0)( -3.0,  0.0)( 11.0,  0.0)

    INPUT MATRIX B ( REAL , IMAGINARY )

(  8.0,  0.0)(  3.0,  0.0)(  1.0,  2.0)( -1.0,  2.0)
(  3.0,  0.0)(  9.0,  0.0)(  1.0, -2.0)( -1.0, -2.0)
(  1.0, -2.0)(  1.0,  2.0)( 10.0,  0.0)( -3.0,  0.0)
( -1.0, -2.0)( -1.0,  2.0)( -3.0,  0.0)( 11.0,  0.0)

*** OUTPUT ***

     IERR =    0

             EIGENVALUE                       EIGENVALUE
             2.3087618D-01                    1.0000000D+00

             EIGENVECTOR                      EIGENVECTOR
       0.17507548 ,  0.00103858         0.20825807 ,  0.00000000
      -0.16011650 ,  0.00086549         0.20825807 ,  0.00000000
      -0.00110807 , -0.14886363         0.00144123 , -0.00000000
       0.00110807 , -0.13795850        -0.00144123 , -0.00000000

             EIGENVALUE                       EIGENVALUE
             1.0000000D+00                    4.3313260D+00

             EIGENVECTOR                      EIGENVECTOR
       0.00314584 , -0.03530977         0.36436426 , -0.00216148
       0.00314584 , -0.03530977        -0.33323187 , -0.00180124
      -0.01730214 ,  0.19420373        -0.00230610 ,  0.30981257
       0.01730214 , -0.19420373         0.00230610 ,  0.28711700
```

4.17.2 ZCGRAN, CCGRAN All Eigenvalues of Hermitian Matrices (Generalized Eigenvalue Problem $Az = \lambda Bz$, B: Positive)

(1) **Function**

Generalized eigenvalue problem

$$A\boldsymbol{z} = \lambda B\boldsymbol{z}$$

(A: Hermitian, B: Positive Hermitian) is solved by using the Cholesky method, the Householder method and QR method to obtain all eigenvalues λ .

(2) **Usage**

Double precision:

CALL ZCGRAN (AR, AI, LNA, N, BR, BI, LNB, E, WORK, IERR)

Single precision:

CALL CCGRAN (AR, AI, LNA, N, BR, BI, LNB, E, WORK, IERR)

(3) **Arguments**

D:Double precision real Z:Double precision complex I: $\left\{\begin{array}{l}\text{INTEGER(4) as for 32bit Integer}\\\text{INTEGER(8) as for 64bit Integer}\end{array}\right\}$
R:Single precision real C:Single precision complex

No.	Argument	Type	Size	Input/ Output	Contents
1	AR	$\left\{\begin{array}{c}\text{D}\\\text{R}\end{array}\right\}$	LNA, N	Input	Real part of Hermitian matrix A
				Output	Input-time contents are not retained.
2	AI	$\left\{\begin{array}{c}\text{D}\\\text{R}\end{array}\right\}$	LNA, N	Input	Imaginary part of Hermitian matrix A
				Output	Input-time contents are not retained.
3	LNA	I	1	Input	Adjustable dimension of arrays AR and AI
4	N	I	1	Input	Order of matrices A and B
5	BR	$\left\{\begin{array}{c}\text{D}\\\text{R}\end{array}\right\}$	LNB, N	Input	Real part of Hermitian matrix B
				Output	Input-time contents are not retained.
6	BI	$\left\{\begin{array}{c}\text{D}\\\text{R}\end{array}\right\}$	LNB, N	Input	Imaginary part of Hermitian matrix B
				Output	Input-time contents are not retained.
7	LNB	I	1	Input	Adjustable dimension of arrays BR and BI

No.	Argument	Type	Size	Input/ Output	Contents
8	E	$\begin{Bmatrix} \mathrm{D} \\ \mathrm{R} \end{Bmatrix}$	N	Output	Eigenvalues λ
9	WORK	$\begin{Bmatrix} \mathrm{D} \\ \mathrm{R} \end{Bmatrix}$	$4 \times \mathrm{N}$	Work	Work area
10	IERR	I	1	Output	Error indicator

(4) **Restrictions**

(a) $1 \leq \mathrm{N} \leq \mathrm{LNA}, \mathrm{LNB}$

(5) **Error indicator**

IERR value	Meaning	Processing
0	Normal termination.	
1000	N was equal to 1.	$\mathrm{E}(1) \leftarrow \mathrm{AR}(1,1)/\mathrm{BR}(1,1)$ is performed.
3000	Restriction (a) was not satisfied.	Processing is aborted.
4000	B was not positive definite.	
5000	The sequence did not converge in the step where the eigenvalue was obtained.	

(6) **Notes**

(a) Arrays AR, AI, BR and BI should be stored only in the upper triangular portions.

(b) Eigenvalues are stored in ascending order.

(c) 4.17.1 $\begin{Bmatrix} \mathrm{ZCGRAA} \\ \mathrm{CCGRAA} \end{Bmatrix}$ should be used if the eigenvectors are needed.

4.18 GENERALIZED EIGENVALUE PROBLEM ($Az = \lambda Bz$) FOR HERMITIAN MATRICES (TWO-DIMENSIONAL ARRAY TYPE) (UPPER TRIANGULAR TYPE) (COMPLEX ARGUMENT TYPE)

4.18.1 ZCGHAA, CCGHAA

All Eigenvalues and All Eigenvectors of Hermitian Matrices (Generalized Eigenvalue Problem $Az = B\lambda z$, B: Positive)

(1) **Function**

Generalized eigenvalue problem

$$A\boldsymbol{z} = \lambda B\boldsymbol{z}$$

(A: Hermitian, B: Positive Hermitian) is solved by using the Cholesky method, the Householder method and QR method to obtain all eigenvalues λ and corresponding all eigenvectors $\boldsymbol{z}$.

(2) **Usage**

Double precision:

CALL ZCGHAA (A, LNA, N, B, LNB, E, WORK, ZZW, IERR)

Single precision:

CALL CCGHAA (A, LNA, N, B, LNB, E, WORK, ZZW, IERR)

(3) **Arguments**

D:Double precision real Z:Double precision complex I: $\left\{\begin{array}{l}\text{INTEGER(4) as for 32bit Integer}\\\text{INTEGER(8) as for 64bit Integer}\end{array}\right\}$

R:Single precision real C:Single precision complex

No.	Argument	Type	Size	Input/ Output	Contents
1	A	$\left\{\begin{array}{c}\text{Z}\\\text{C}\end{array}\right\}$	LNA, N	Input	Hermitian matrix A
				Output	Eigenvector z
2	LNA	I	1	Input	Adjustable dimension of array A
3	N	I	1	Input	Order of matrices A and B
4	B	$\left\{\begin{array}{c}\text{Z}\\\text{C}\end{array}\right\}$	LNB, N	Input	Hermitian matrix B
				Output	Input-time contents are not retained.
5	LNB	I	1	Input	Adjustable dimension of array B
6	E	$\left\{\begin{array}{c}\text{D}\\\text{R}\end{array}\right\}$	N	Output	Eigenvalues λ

No.	Argument	Type	Size	Input/ Output	Contents
7	WORK	$\begin{Bmatrix} \text{D} \\ \text{R} \end{Bmatrix}$	2 × N	Work	Work area
8	ZZW	$\begin{Bmatrix} \text{Z} \\ \text{C} \end{Bmatrix}$	N	Work	Work area
9	IERR	I	1	Output	Error indicator

(4) **Restrictions**

(a) $1 \leq \text{N} \leq \text{LNA}, \text{LNB}$

(5) **Error indicator**

IERR value	Meaning	Processing
0	Normal termination.	
1000	N was equal to 1.	$\text{E}(1) \leftarrow \text{A}(1,1)/\text{B}(1,1)$ and $\text{A}(1,1) \leftarrow 1.0/\sqrt{\text{B}(1,1)}$ are performed.
3000	Restriction (a) was not satisfied.	Processing is aborted.
4000	B was not positive definite.	
5000	The sequence did not converge in the step where the eigenvalue was obtained.	

(6) **Notes**

(a) Arrays A and B should be stored only in the upper triangular portions.

(b) Eigenvalues are stored in ascending order.

(c) Eigenvectors $\boldsymbol{v}_i$ are an orthonormal set so that $\boldsymbol{v}_j^* B \boldsymbol{v}_k = \delta_{j,k}$

(d) 4.18.2 $\begin{Bmatrix} \text{ZCGHAN} \\ \text{CCGHAN} \end{Bmatrix}$ should be used if the eigenvectors are not needed.

(7) **Example**

(a) Problem
For Hermitian matrix of the degree 4

$$A = \begin{bmatrix} 8 & 3 & 1-2i & -1-2i \\ 3 & 9 & 1+2i & -1+2i \\ 1+2i & 1-2i & 10 & -3 \\ -1+2i & -1-2i & -3 & 11 \end{bmatrix}$$

and its conjugate Hermitian matrix

$$B = \begin{bmatrix} 8 & 3 & 1+2i & -1+2i \\ 3 & 9 & 1-2i & -1-2i \\ 1-2i & 1+2i & 10 & -3 \\ -1-2i & -1+2i & -3 & 11 \end{bmatrix}$$

obtain eigenvector of generalized eigenvalue problem.

(b) Input data

N=4, LNA=4, matrix A, LNB=4 and matrix B.

(c) Main program

```
      PROGRAM ACGHAA
      IMPLICIT REAL(8)(A-H,O-Z)
      PARAMETER ( N=4 , LN=4 )
      REAL(8)      E(N), WORK(2*N)
      COMPLEX(8)  KEEP(LN,N),VM,A(LN,N),B(LN,N),ZZW(N)
      REAL(8)      AR(LN,N),BR(LN,N),AI(LN,N),BI(LN,N),SCL(N)
      KEEP(1,1)=( 8.D0, 0.D0)
      KEEP(2,2)=( 9.D0, 0.D0)
      KEEP(3,3)=( 10.D0, 0.D0)
      KEEP(4,4)=( 11.D0, 0.D0)
      KEEP(1,2)=( 3.D0, 0.D0)
      KEEP(1,3)=( 1.D0, 2.D0)
      KEEP(1,4)=(-1.D0, 2.D0)
      KEEP(2,3)=( 1.D0,-2.D0)
      KEEP(2,4)=(-1.D0,-2.D0)
      KEEP(3,4)=(-3.D0, 0.D0)
      METHOD=0
      WRITE(6,60)
      IF(METHOD.EQ.0) WRITE(6,70) N
      DO 1000 I=1,N
      DO 1001 J=I,N
        BR(I,J)=DBLE (KEEP(I,J))
        BI(I,J)=DIMAG(KEEP(I,J))
        KEEP(J,I)=CONJG(KEEP(I,J))
        AR(I,J)=BR(I,J)
        AI(I,J)=-BI(I,J)
        AR(J,I)=AR(I,J)
        AI(J,I)=-AI(I,J)
        BR(J,I)=BR(I,J)
        BI(J,I)=-BI(I,J)
 1001 CONTINUE
 1000 CONTINUE
      IF(METHOD.EQ.0) THEN
         WRITE(6,80)
         DO 1150 I=1,N
         WRITE(6,5000) AR(I,1),AI(I,1),AR(I,2),AI(I,2),&
                       AR(I,3),AI(I,3),AR(I,4),AI(I,4)
 1150    CONTINUE
         WRITE(6,90)
         DO 1250 I=1,N
         WRITE(6,5000) BR(I,1),BI(I,1),BR(I,2),BI(I,2),&
                       BR(I,3),BI(I,3),BR(I,4),BI(I,4)
 1250    CONTINUE
      ENDIF
      IF(METHOD.GT.0) THEN
         CALL BWINTA(AR, AI, BR, BI, LN, N, SCL, IERRA)
         IF(IERRA.GT.0) STOP
         WRITE(6,95)
         DO 1100 I=1,N
         DO 1101 J=I,N
            WRITE(6,6000) AR(I,J),AI(I,J),BR(I,J),BI(I,J),I,J
 1101    CONTINUE
 1100    CONTINUE
      ENDIF
      DO 1200  I=1,N
      DO 1201  J=I,N
         A(I,J)=CMPLX(AR(I,J),AI(I,J), KIND=8)
         B(I,J)=CMPLX(BR(I,J),BI(I,J), KIND=8)
 1201 CONTINUE
 1200 CONTINUE
!
      CALL ZCGHAA(A, LN, N, B, LN, E, WORK, ZZW, IERR)
!
      DO 1300  I=1,N
      DO 1301  J=1,N
         AR(I,J)=A(I,J)
         AI(I,J)=DIMAG(A(I,J))
 1301 CONTINUE
 1300 CONTINUE
      IF(METHOD.GT.0) THEN
         CALL BWINTB(AR, AI, LN, N, SCL, IERRB)
         IF(IERRB.GT.0) STOP
      ENDIF
      IF(METHOD.GT.0) THEN
         WRITE(6,96)
      ELSE
         WRITE(6,97)
      ENDIF
      WRITE(6,98) IERR
      DO 2000 I=1,N,2
         VM=(0.D0,0.D0)
         DO 2100 J=1,N
         DO 2101 K=1,N
            VM=VM + KEEP(J,K)*CMPLX(AR(K,I), AI(K,I), KIND=8)&
                      *CMPLX(AR(J,I),-AI(J,I), KIND=8)
 2101    CONTINUE
 2100    CONTINUE
         WRITE(6,6500) ('EIGENVALUE ',L=1,2)
         WRITE(6,7000) E(I),E(I+1)
         WRITE(6,6500) ('EIGENVECTOR',L=1,2)
         WRITE(6,8000) AR(1,I),AI(1,I),AR(1,I+1),AI(1,I+1)
         WRITE(6,8000) AR(2,I),AI(2,I),AR(2,I+1),AI(2,I+1)
         WRITE(6,8000) AR(3,I),AI(3,I),AR(3,I+1),AI(3,I+1)
         WRITE(6,8000) AR(4,I),AI(4,I),AR(4,I+1),AI(4,I+1)
 2000 CONTINUE
      STOP
   60 FORMAT(1X,  ' *** ZCGHAA *** ',/,/)
   70 FORMAT(1X, ' *** INPUT ***',/,/,'     N= ',I3,/,/)
   80 FORMAT(1X,'     INPUT MATRIX A ( REAL , IMAGINARY )',/)
   90 FORMAT(1X,/,/,'     INPUT MATRIX B ( REAL , IMAGINARY )',/)
   95 FORMAT(1X,' *** INPUT *** (SCALED) ')
   96 FORMAT(1X,' *** OUTPUT *** (SCALED) ')
   97 FORMAT(1X,/,/, ' *** OUTPUT ***',/,/)
   98 FORMAT(1X,'     IERR = ',I4)
 5000 FORMAT(1X,4('(',F5.1,',',F5.1,')'))
 6000 FORMAT(1X,' RE(A) = ',E10.2,' IM(A) = ',E10.2,&
                ' RE(B) = ',E10.2,' IM(B) = ',E10.2,&
                 ' I,J = ',I2,3X,I2)
 6500 FORMAT(1X ,/, 2(14X,A11,8X))
 7000 FORMAT(1X,2(12X,1PD14.7,7X))
 8000 FORMAT(1X, 2(5X,F12.8,' ,',F12.8,2X))
      END
      SUBROUTINE BWINTA(AR, AI, BR, BI, LN, N, SCL, IERR)
      INTEGER           LN, N
      REAL(8)  AR(LN, N), BR(LN, N), SCL(N),&
                        AI(LN, N), BI(LN, N)
!
      INTEGER I,J
      REAL(8) F,ZERO,ONE
      PARAMETER(ZERO=0.D0)
      PARAMETER(ONE =1.D0)
!
      IERR=0
      IF(N.GT.LN) IERR=3000
      IF(N.LT. 1) IERR=3000
      IF(IERR.GT.0) RETURN
      DO 1000 I=1,N
         IF(BR(I,I).LE.ZERO) THEN
              IERR=4000
              RETURN
         ENDIF
         SCL(I)=ONE/SQRT(BR(I,I))
 1000 CONTINUE
      DO 2000 J=1,N
      DO 2001 I=1,N
         F=SCL(I)*SCL(J)
         AR(I,J)=AR(I,J)*F
         AI(I,J)=AI(I,J)*F
         BR(I,J)=BR(I,J)*F
         BI(I,J)=BI(I,J)*F
 2001 CONTINUE
 2000 CONTINUE
!
      RETURN
      END
!
      SUBROUTINE BWINTB(AR, AI, LN, N, SCL, IERR)
      INTEGER           LN, N
      REAL(8)  AR(LN, N), AI(LN, N), SCL(N)
      INTEGER I,J
!
      IERR=0
      IF(N.GT.LN) IERR=3000
      IF(N.LT. 1) IERR=3000
      IF(IERR.GT.0) RETURN
!
      DO 1000 I=1,N
      DO 1001 J=1,N
         AR(I,J)=AR(I,J)*SCL(I)
         AI(I,J)=AI(I,J)*SCL(I)
 1001 CONTINUE
 1000 CONTINUE
!
      RETURN
      END
```

(d) Output results

```
 *** ZCGHAA ***


 *** INPUT ***

    N=   4


     INPUT MATRIX A ( REAL , IMAGINARY )

(  8.0,  0.0)(  3.0,  0.0)(  1.0, -2.0)( -1.0, -2.0)
(  3.0,  0.0)(  9.0,  0.0)(  1.0,  2.0)( -1.0,  2.0)
(  1.0,  2.0)(  1.0, -2.0)( 10.0,  0.0)( -3.0,  0.0)
( -1.0,  2.0)( -1.0, -2.0)( -3.0,  0.0)( 11.0,  0.0)
```

```
    INPUT MATRIX B ( REAL , IMAGINARY )

(  8.0,  0.0)(  3.0,  0.0)(  1.0,  2.0)( -1.0,  2.0)
(  3.0,  0.0)(  9.0,  0.0)(  1.0, -2.0)( -1.0, -2.0)
(  1.0, -2.0)(  1.0,  2.0)( 10.0,  0.0)( -3.0,  0.0)
( -1.0, -2.0)( -1.0,  2.0)( -3.0,  0.0)( 11.0,  0.0)

*** OUTPUT ***

     IERR =    0
             EIGENVALUE                        EIGENVALUE
             2.3087618D-01                     1.0000000D+00

             EIGENVECTOR                       EIGENVECTOR
       0.17507548 ,  0.00103858          0.20825807 ,  0.00000000
      -0.16011650 ,  0.00086549          0.20825807 ,  0.00000000
      -0.00110807 , -0.14886363          0.00144123 , -0.00000000
       0.00110807 , -0.13795850         -0.00144123 , -0.00000000

             EIGENVALUE                        EIGENVALUE
             1.0000000D+00                     4.3313260D+00

             EIGENVECTOR                       EIGENVECTOR
       0.00314584 , -0.03530977          0.36436426 , -0.00216148
       0.00314584 , -0.03530977         -0.33323187 , -0.00180124
      -0.01730214 ,  0.19420373         -0.00230610 ,  0.30981257
       0.01730214 , -0.19420373          0.00230610 ,  0.28711700
```

4.18.2 ZCGHAN, CCGHAN All Eigenvalues of Hermitian Matrices (Generalized Eigenvalue Problem $Az = B\lambda z$, B: Positive)

(1) **Function**

Generalized eigenvalue problem

$$A\boldsymbol{z} = \lambda B\boldsymbol{z}$$

(A: Hermitian, B: Positive Hermitian) is solved by using the Cholesky method, the Householder method and QR method to obtain all eigenvalues λ .

(2) **Usage**

Double precision:

CALL ZCGHAN (A, LNA, N, B, LNB, E, WORK, ZZW, IERR)

Single precision:

CALL CCGHAN (A, LNA, N, B, LNB, E, WORK, ZZW, IERR)

(3) **Arguments**

D:Double precision real Z:Double precision complex
R:Single precision real C:Single precision complex
I: $\left\{\begin{matrix}\text{INTEGER(4) as for 32bit Integer}\\ \text{INTEGER(8) as for 64bit Integer}\end{matrix}\right\}$

No.	Argument	Type	Size	Input/ Output	Contents
1	A	$\left\{\begin{matrix}Z\\C\end{matrix}\right\}$	LNA, N	Input	Hermitian matrix A
				Output	Input-time contents are not retained.
2	LNA	I	1	Input	Adjustable dimension of array A
3	N	I	1	Input	Order of matrices A and B
4	B	$\left\{\begin{matrix}Z\\C\end{matrix}\right\}$	LNB, N	Input	Hermitian matrix B
				Output	Input-time contents are not retained.
5	LNB	I	1	Input	Adjustable dimension of array B
6	E	$\left\{\begin{matrix}D\\R\end{matrix}\right\}$	N	Output	Eigenvalues λ
7	WORK	$\left\{\begin{matrix}D\\R\end{matrix}\right\}$	2 × N	Work	Work area
8	ZZW	$\left\{\begin{matrix}Z\\C\end{matrix}\right\}$	N	Work	Work area
9	IERR	I	1	Output	Error indicator

(4) **Restrictions**

(a) $1 \leq \mathrm{N} \leq \mathrm{LNA}, \mathrm{LNB}$

(5) **Error indicator**

IERR value	Meaning	Processing
0	Normal termination.	
1000	N was equal to 1.	$\mathrm{E}(1) \leftarrow \mathrm{A}(1,1)/\mathrm{B}(1,1)$ is performed.
3000	Restriction (a) was not satisfied.	Processing is aborted.
4000	*B* was not positive definite.	
5000	The sequence did not converge in the step where the eigenvalue was obtained.	

(6) **Notes**

(a) Arrays A and B should be stored only in the upper triangular portions.

(b) Eigenvalues are stored in ascending order.

(c) 4.18.1 $\left\{\begin{matrix}\mathrm{ZCGHAA}\\ \mathrm{CCGHAA}\end{matrix}\right\}$ should be used if the eigenvectors are needed.

4.19 GENERALIZED EIGENVALUE PROBLEM ($ABz = \lambda z$) FOR HERMITIAN MATRICES (TWO-DIMENSIONAL ARRAY TYPE) (UPPER TRIANGULAR TYPE) (REAL ARGUMENT TYPE)

4.19.1 ZCGJAA, CCGJAA
All Eigenvalues and All Eigenvectors of Hermitian Matrices (Generalized Eigenvalue Problem $ABz = \lambda z$, B: Positive)

(1) **Function**

Generalized eigenvalue problem

$$AB\boldsymbol{z} = \lambda \boldsymbol{z}$$

(A: Hermitian, B: Positive Hermitian) is solved by using the Cholesky method, the Householder method and QR method to obtain all eigenvalues λ and corresponding all eigenvectors $\boldsymbol{z}$.

(2) **Usage**

Double precision:

CALL ZCGJAA (AR, AI, LNA, N, BR, BI, LNB, E, WORK, IERR)

Single precision:

CALL CCGJAA (AR, AI, LNA, N, BR, BI, LNB, E, WORK, IERR)

(3) **Arguments**

D:Double precision real Z:Double precision complex I: $\left\{\begin{array}{l}\text{INTEGER(4) as for 32bit Integer}\\ \text{INTEGER(8) as for 64bit Integer}\end{array}\right\}$
R:Single precision real C:Single precision complex

No.	Argument	Type	Size	Input/ Output	Contents
1	AR	$\left\{\begin{matrix}\text{D}\\ \text{R}\end{matrix}\right\}$	LNA, N	Input	Real part of Hermitian matrix A
				Output	Real part of eigenvector z
2	AI	$\left\{\begin{matrix}\text{D}\\ \text{R}\end{matrix}\right\}$	LNA, N	Input	Imaginary part of Hermitian matrix A
				Output	Imaginary part of eigenvector z
3	LNA	I	1	Input	Adjustable dimension of arrays AR and AI
4	N	I	1	Input	Order of matrices A and B
5	BR	$\left\{\begin{matrix}\text{D}\\ \text{R}\end{matrix}\right\}$	LNB, N	Input	Real part of Hermitian matrix B
				Output	Input-time contents are not retained.
6	BI	$\left\{\begin{matrix}\text{D}\\ \text{R}\end{matrix}\right\}$	LNB, N	Input	Imaginary part of Hermitian matrix B
				Output	Input-time contents are not retained.

No.	Argument	Type	Size	Input/ Output	Contents
7	LNB	I	1	Input	Adjustable dimension of arrays BR and BI
8	E	$\begin{Bmatrix}\text{D}\\\text{R}\end{Bmatrix}$	N	Output	Eigenvalues λ
9	WORK	$\begin{Bmatrix}\text{D}\\\text{R}\end{Bmatrix}$	$4 \times \text{N}$	Work	Work area
10	IERR	I	1	Output	Error indicator

(4) **Restrictions**

(a) $1 \leq \text{N} \leq \text{LNA}, \text{LNB}$

(5) **Error indicator**

IERR value	Meaning	Processing
0	Normal termination.	
1000	N was equal to 1.	$\text{E}(1) \leftarrow \text{A}(1,1) \times \text{B}(1,1)$ and $\text{A}(1,1) \leftarrow 1.0/\sqrt{\text{B}(1,1)}$ are performed.
3000	Restriction (a) was not satisfied.	Processing is aborted.
4000	B was not positive definite.	
5000	The sequence did not converge in the step where the eigenvalue was obtained.	

(6) **Notes**

(a) Arrays AR, AI, BR and BI should be stored only in the upper triangular portions.

(b) Eigenvalues are stored in ascending order.

(c) Eigenvectors $\boldsymbol{v}_i$ are an orthonormal set so that $\boldsymbol{v}_j^* B \boldsymbol{v}_k = \delta_{j,k}$

(d) 4.19.2 $\begin{Bmatrix}\text{ZCGJAN}\\\text{CCGJAN}\end{Bmatrix}$ should be used if the eigenvectors are not needed.

(e) 4.20.1 $\begin{Bmatrix}\text{ZCGKAA}\\\text{CCGKAA}\end{Bmatrix}$ should be used if matrix A is only positive.

(7) **Example**

(a) Problem
For Hermitian matrix of the degree 4

$$A = \begin{bmatrix} 8 & 3 & 1-2i & -1-2i \\ 3 & 9 & 1+2i & -1+2i \\ 1+2i & 1-2i & 10 & -3 \\ -1+2i & -1-2i & -3 & 11 \end{bmatrix}$$

and its conjugate Hermitian matrix

$$B = \begin{bmatrix} 8 & 3 & 1+2i & -1+2i \\ 3 & 9 & 1-2i & -1-2i \\ 1-2i & 1+2i & 10 & -3 \\ -1-2i & -1+2i & -3 & 11 \end{bmatrix}$$

obtain eigenvector of generalized eigenvalue problem $ABz = \lambda z$.

(b) Input data

N=4, LNA=4, matrix A, LNB=4 and matrix B.

(c) Main program

```
      PROGRAM ACGJAA
! *** EXAMPLE OF ZCGJAA ***
      IMPLICIT NONE
!
      INTEGER N,LNA,LNB
      PARAMETER( N = 4, LNA = 4, LNB = 4 )
      INTEGER IERR,I,J,L
      REAL(8) AR(LNA,N),BR(LNB,N),AI(LNA,N),BI(LNB,N)
      REAL(8) E(N),WORK(4*N)
      COMPLEX(8) KEEP(LNA,N)
!
      KEEP(1,1)=( 8.D0, 0.D0)
      KEEP(2,2)=( 9.D0, 0.D0)
      KEEP(3,3)=( 10.D0, 0.D0)
      KEEP(4,4)=( 11.D0, 0.D0)
      KEEP(1,2)=( 3.D0, 0.D0)
      KEEP(1,3)=( 1.D0, 2.D0)
      KEEP(1,4)=(-1.D0, 2.D0)
      KEEP(2,3)=( 1.D0,-2.D0)
      KEEP(2,4)=(-1.D0,-2.D0)
      KEEP(3,4)=(-3.D0, 0.D0)
      WRITE(6,6000) N, LNA, LNB
      DO 100 I=1,N
      DO 110 J=I,N
         BR(I,J)=DBLE (KEEP(I,J))
         BI(I,J)=DIMAG(KEEP(I,J))
         KEEP(J,I)=CONJG(KEEP(I,J))
         AR(I,J)=BR(I,J)
         AI(I,J)=-BI(I,J)
         AR(J,I)=AR(I,J)
         AI(J,I)=-AI(I,J)
         BR(J,I)=BR(I,J)
         BI(J,I)=-BI(I,J)
  110 CONTINUE
  100 CONTINUE
      WRITE(6,6010)
      DO 120 I=1,N
         WRITE(6,6020) AR(I,1),AI(I,1),AR(I,2),AI(I,2),&
                       AR(I,3),AI(I,3),AR(I,4),AI(I,4)
  120 CONTINUE
      WRITE(6,6030)
      DO 130 I=1,N
         WRITE(6,6020) BR(I,1),BI(I,1),BR(I,2),BI(I,2),&
                       BR(I,3),BI(I,3),BR(I,4),BI(I,4)
  130 CONTINUE
!
      CALL ZCGJAA(AR,AI, LNA, N, BR,BI, LNB, E, WORK, IERR)
!
      WRITE(6,6040) IERR
      DO 140 I=1,N,2
         WRITE(6,6050) (' EIGENVALUE',L=1,2)
         WRITE(6,6060) E(I),E(I+1)
         WRITE(6,6050) ('EIGENVECTOR',L=1,2)
         WRITE(6,6070) AR(1,I),AI(1,I),AR(1,I+1),AI(1,I+1)
         WRITE(6,6070) AR(2,I),AI(2,I),AR(2,I+1),AI(2,I+1)
         WRITE(6,6070) AR(3,I),AI(3,I),AR(3,I+1),AI(3,I+1)
         WRITE(6,6070) AR(4,I),AI(4,I),AR(4,I+1),AI(4,I+1)
  140 CONTINUE
!
      STOP
 6000 FORMAT(/,&
             1X,'***  ZCGJAA  ***',/,/,&
             1X,' **  INPUT  **',/,/,&
             1X,'     N = ',I4, '    LNA = ',I4, '   LNB = ',I4,/)
 6010 FORMAT(/,&
             1X,'     INPUT MATRIX A ( REAL , IMAGINARY )',/)
 6020 FORMAT(1X,5X,4('(',F5.1,',',F5.1,')'))
 6030 FORMAT(/,&
             1X,'     INPUT MATRIX B ( REAL , IMAGINARY )',/)
 6040 FORMAT(/,&
             1X,' **  OUTPUT  **',/,/,&
             1X,'     IERR =',I5,/)
 6050 FORMAT(/,&
             1X,14X,A11,22X,A11)
 6060 FORMAT(1X,12X,1PD14.7,19X,1PD14.7)
 6070 FORMAT(1X,5X,F12.8,’ ,’,F12.8,7X,F12.8,’ ,’,F12.8)
 6080 FORMAT(1X,’ RE(A) = ’,D10.2,’ IM(A) = ’,D10.2,&
                ’ RE(B) = ’,D10.2,’ IM(B) = ’,D10.2,&
                ’ I,J = ’,I2,3X,I2)
      END
```

(d) Output results

```
***  ZCGJAA  ***

 **  INPUT  **

     N =    4    LNA =    4   LNB =    4

     INPUT MATRIX A ( REAL , IMAGINARY )

     (  8.0,  0.0)(  3.0,  0.0)(  1.0, -2.0)( -1.0, -2.0)
     (  3.0,  0.0)(  9.0,  0.0)(  1.0,  2.0)( -1.0,  2.0)
     (  1.0,  2.0)(  1.0, -2.0)( 10.0,  0.0)( -3.0,  0.0)
     ( -1.0,  2.0)( -1.0, -2.0)( -3.0,  0.0)( 11.0,  0.0)

     INPUT MATRIX B ( REAL , IMAGINARY )

     (  8.0,  0.0)(  3.0,  0.0)(  1.0,  2.0)( -1.0,  2.0)
     (  3.0,  0.0)(  9.0,  0.0)(  1.0, -2.0)( -1.0, -2.0)
     (  1.0, -2.0)(  1.0,  2.0)( 10.0,  0.0)( -3.0,  0.0)
     ( -1.0, -2.0)( -1.0,  2.0)( -3.0,  0.0)( 11.0,  0.0)

 **  OUTPUT  **

     IERR =    0


               EIGENVALUE                       EIGENVALUE
             1.6653903D+01                    3.6956492D+01

              EIGENVECTOR                      EIGENVECTOR
      -0.39854844 , -0.00036439         0.10790304 , -0.01173201
       0.34532368 ,  0.00125155        -0.10272938 , -0.00491675
       0.00738031 , -0.13233200         0.01670018 ,  0.32687634
      -0.00409275 , -0.12491736         0.01489428 ,  0.28145894

               EIGENVALUE                       EIGENVALUE
             1.0637116D+02                    2.1801844D+02

              EIGENVECTOR                      EIGENVECTOR
       0.17017623 , -0.00775370         0.09159450 ,  0.00390728
       0.20301495 , -0.00090493         0.10133798 ,  0.00028167
      -0.09985610 , -0.00146802         0.14661381 ,  0.00223902
       0.12968534 , -0.00655395        -0.16597958 , -0.00438924
```

4.19.2 ZCGJAN, CCGJAN All Eigenvalues of Hermitian Matrices (Generalized Eigenvalue Problem $ABz = \lambda z$, B: Positive)

(1) **Function**

Generalized eigenvalue problem

$$A B \boldsymbol{z} = \lambda \boldsymbol{z}$$

(A: Hermitian, B: Positive Hermitian) is solved by using the Cholesky method, the Householder method and QR method to obtain all eigenvalues λ .

(2) **Usage**

Double precision:

CALL ZCGJAN (AR, AI, LNA, N, BR, BI, LNB, E, WORK, IERR)

Single precision:

CALL CCGJAN (AR, AI, LNA, N, BR, BI, LNB, E, WORK, IERR)

(3) **Arguments**

D:Double precision real Z:Double precision complex I: $\left\{ \begin{array}{l} \text{INTEGER(4) as for 32bit Integer} \\ \text{INTEGER(8) as for 64bit Integer} \end{array} \right\}$
R:Single precision real C:Single precision complex

No.	Argument	Type	Size	Input/ Output	Contents
1	AR	$\left\{\begin{array}{c}\text{D}\\\text{R}\end{array}\right\}$	LNA, N	Input	Real part of Hermitian matrix A
				Output	Input-time contents are not retained.
2	AI	$\left\{\begin{array}{c}\text{D}\\\text{R}\end{array}\right\}$	LNA, N	Input	Imaginary part of Hermitian matrix A
				Output	Input-time contents are not retained.
3	LNA	I	1	Input	Adjustable dimension of arrays AR and AI
4	N	I	1	Input	Order of matrices A and B
5	BR	$\left\{\begin{array}{c}\text{D}\\\text{R}\end{array}\right\}$	LNB, N	Input	Real part of Hermitian matrix B
				Output	Input-time contents are not retained.
6	BI	$\left\{\begin{array}{c}\text{D}\\\text{R}\end{array}\right\}$	LNB, N	Input	Imaginary part of Hermitian matrix B
				Output	Input-time contents are not retained.
7	LNB	I	1	Input	Adjustable dimension of arrays BR and BI

No.	Argument	Type	Size	Input/ Output	Contents
8	E	$\begin{Bmatrix} \mathrm{D} \\ \mathrm{R} \end{Bmatrix}$	N	Output	Eigenvalues λ
9	WORK	$\begin{Bmatrix} \mathrm{D} \\ \mathrm{R} \end{Bmatrix}$	4 × N	Work	Work area
10	IERR	I	1	Output	Error indicator

(4) **Restrictions**

(a) $1 \le \mathrm{N} \le \mathrm{LNA}, \mathrm{LNB}$

(5) **Error indicator**

IERR value	Meaning	Processing
0	Normal termination.	
1000	N was equal to 1.	$\mathrm{E}(1) \leftarrow \mathrm{A}(1,1) \times \mathrm{B}(1,1)$ is performed.
3000	Restriction (a) was not satisfied.	Processing is aborted.
4000	B was not positive definite.	
5000	The sequence did not converge in the step where the eigenvalue was obtained.	

(6) **Notes**

(a) Arrays AR, AI, BR and BI should be stored only in the upper triangular portions.

(b) Eigenvalues are stored in ascending order.

(c) 4.19.1 $\begin{Bmatrix} \text{ZCGJAA} \\ \text{CCGJAA} \end{Bmatrix}$ should be used if the eigenvectors are needed.

(d) 4.20.2 $\begin{Bmatrix} \text{ZCGKAN} \\ \text{CCGKAN} \end{Bmatrix}$ should be used if matrix A is only positive.

4.20 GENERALIZED EIGENVALUE PROBLEM ($BAz = \lambda z$) FOR HERMITIAN MATRICES (TWO-DIMENSIONAL ARRAY TYPE) (UPPER TRIANGULAR TYPE) (REAL ARGUMENT TYPE)

4.20.1 ZCGKAA, CCGKAA
All Eigenvalues and All Eigenvectors of Hermitian Matrices (Generalized Eigenvalue Problem $BAz = \lambda z$, B: Positive)

(1) **Function**

Generalized eigenvalue problem

$$BA\boldsymbol{z} = \lambda \boldsymbol{z}$$

(A: Hermitian, B: Positive Hermitian) is solved by using the Cholesky method, the Householder method and QR method to obtain all eigenvalues λ and corresponding all eigenvectors $\boldsymbol{z}$.

(2) **Usage**

Double precision:

CALL ZCGKAA (AR, AI, LNA, N, BR, BI, LNB, E, WORK, IERR)

Single precision:

CALL CCGKAA (AR, AI, LNA, N, BR, BI, LNB, E, WORK, IERR)

(3) **Arguments**

D:Double precision real Z:Double precision complex I: $\left\{ \begin{array}{l} \text{INTEGER(4) as for 32bit Integer} \\ \text{INTEGER(8) as for 64bit Integer} \end{array} \right\}$

R:Single precision real C:Single precision complex

No.	Argument	Type	Size	Input/ Output	Contents
1	AR	$\begin{Bmatrix} \text{D} \\ \text{R} \end{Bmatrix}$	LNA, N	Input	Real part of Hermitian matrix A
				Output	Real part of eigenvector z
2	AI	$\begin{Bmatrix} \text{D} \\ \text{R} \end{Bmatrix}$	LNA, N	Input	Imaginary part of Hermitian matrix A
				Output	Imaginary part of eigenvector z
3	LNA	I	1	Input	Adjustable dimension of arrays AR and AI
4	N	I	1	Input	Order of matrices A and B
5	BR	$\begin{Bmatrix} \text{D} \\ \text{R} \end{Bmatrix}$	LNB, N	Input	Real part of Hermitian matrix B
				Output	Input-time contents are not retained.

No.	Argument	Type	Size	Input/ Output	Contents
6	BI	$\begin{Bmatrix} \text{D} \\ \text{R} \end{Bmatrix}$	LNB, N	Input	Imaginary part of Hermitian matrix B
				Output	Input-time contents are not retained.
7	LNB	I	1	Input	Adjustable dimension of arrays BR and BI
8	E	$\begin{Bmatrix} \text{D} \\ \text{R} \end{Bmatrix}$	N	Output	Eigenvalues λ
9	WORK	$\begin{Bmatrix} \text{D} \\ \text{R} \end{Bmatrix}$	$4 \times \text{N}$	Work	Work area
10	IERR	I	1	Output	Error indicator

(4) **Restrictions**

(a) $1 \le \text{N} \le \text{LNA}, \text{LNB}$

(5) **Error indicator**

IERR value	Meaning	Processing
0	Normal termination.	
1000	N was equal to 1.	$\text{E}(1) \leftarrow \text{A}(1,1) \times \text{B}(1,1)$ and $\text{A}(1,1) \leftarrow \sqrt{\text{B}(1,1)}$ are performed.
3000	Restriction (a) was not satisfied.	Processing is aborted.
4000	B was not positive definite.	
5000	The sequence did not converge in the step where the eigenvalue was obtained.	

(6) **Notes**

(a) Arrays AR, AI, BR and BI should be stored only in the upper triangular portions.

(b) Eigenvalues are stored in ascending order.

(c) Eigenvectors $\boldsymbol{v}_i$ are an orthonormal set so that $\boldsymbol{v}_j^* B^{-1} \boldsymbol{v}_k = \delta_{j,k}$

(d) 4.20.2 $\begin{Bmatrix} \text{ZCGKAN} \\ \text{CCGKAN} \end{Bmatrix}$ should be used if the eigenvectors are not needed.

(e) 4.19.1 $\begin{Bmatrix} \text{ZCGJAA} \\ \text{CCGJAA} \end{Bmatrix}$ should be used if matrix A is only positive.

(7) **Example**

(a) Problem

For Hermitian matrix of the degree 4

$$A = \begin{bmatrix} 8 & 3 & 1-2i & -1-2i \\ 3 & 9 & 1+2i & -1+2i \\ 1+2i & 1-2i & 10 & -3 \\ -1+2i & -1-2i & -3 & 11 \end{bmatrix}$$

and its conjugate Hermitian matrix

$$B = \begin{bmatrix} 8 & 3 & 1+2i & -1+2i \\ 3 & 9 & 1-2i & -1-2i \\ 1-2i & 1+2i & 10 & -3 \\ -1-2i & -1+2i & -3 & 11 \end{bmatrix}$$

obtain eigenvector of generalized eigenvalue problem $BAz = \lambda z$.

(b) Input data

N=4, LNA=4, matrix A, LNB=4 and matrix B.

(c) Main program

```
      PROGRAM ACGKAA
! *** EXAMPLE OF ZCGKAA ***
      IMPLICIT NONE
!
      INTEGER N,LNA,LNB
      PARAMETER( N=4 , LNA=4 , LNB=4 )
      INTEGER IERR,I,J,L
      REAL(8) AR(LNA,N),BR(LNB,N),AI(LNA,N),BI(LNB,N)
      REAL(8) E(N),WORK(4*N)
      COMPLEX(8)  KEEP(LNA,N)
!
      KEEP(1,1)=( 8.D0, 0.D0)
      KEEP(2,2)=( 9.D0, 0.D0)
      KEEP(3,3)=( 10.D0, 0.D0)
      KEEP(4,4)=( 11.D0, 0.D0)
      KEEP(1,2)=( 3.D0, 0.D0)
      KEEP(1,3)=( 1.D0, 2.D0)
      KEEP(1,4)=(-1.D0, 2.D0)
      KEEP(2,3)=( 1.D0,-2.D0)
      KEEP(2,4)=(-1.D0,-2.D0)
      KEEP(3,4)=(-3.D0, 0.D0)
      WRITE(6,6000) N, LNA, LNB
      DO 100 I=1,N
      DO 110 J=I,N
         BR(I,J)=DBLE (KEEP(I,J))
         BI(I,J)=DIMAG(KEEP(I,J))
         KEEP(J,I)=CONJG(KEEP(I,J))
         AR(I,J)=BR(I,J)
         AI(I,J)=-BI(I,J)
         AR(J,I)=AR(I,J)
         AI(J,I)=-AI(I,J)
         BR(J,I)=BR(I,J)
         BI(J,I)=-BI(I,J)
  110 CONTINUE
  100 CONTINUE
      WRITE(6,6010)
      DO 120 I=1,N
         WRITE(6,6020) AR(I,1),AI(I,1),AR(I,2),AI(I,2),&
                       AR(I,3),AI(I,3),AR(I,4),AI(I,4)
  120 CONTINUE
      WRITE(6,6030)
      DO 130 I=1,N
         WRITE(6,6020) BR(I,1),BI(I,1),BR(I,2),BI(I,2),&
                       BR(I,3),BI(I,3),BR(I,4),BI(I,4)
  130 CONTINUE
!
      CALL ZCGKAA(AR,AI, LNA, N, BR,BI, LNB, E, WORK, IERR)
!
      WRITE(6,6040) IERR
      DO 140 I=1,N,2
         WRITE(6,6050) (' EIGENVALUE',L=1,2)
         WRITE(6,6060) E(I),E(I+1)
         WRITE(6,6050) ('EIGENVECTOR',L=1,2)
         WRITE(6,6070) AR(1,I),AI(1,I),AR(1,I+1),AI(1,I+1)
         WRITE(6,6070) AR(2,I),AI(2,I),AR(2,I+1),AI(2,I+1)
         WRITE(6,6070) AR(3,I),AI(3,I),AR(3,I+1),AI(3,I+1)
         WRITE(6,6070) AR(4,I),AI(4,I),AR(4,I+1),AI(4,I+1)
  140 CONTINUE
!
      STOP
 6000 FORMAT(/,&
             1X,'***  ZCGKAA  ***',/,/,&
             1X,' **  INPUT  **',/,/,&
             1X,'     N = ',I4, '    LNA = ',I4, '   LNB = ',I4,/)
 6010 FORMAT(/,&
             1X,'     INPUT MATRIX A ( REAL , IMAGINARY )',/)
 6020 FORMAT(1X,5X,4('(',F5.1,',',F5.1,')'))
 6030 FORMAT(/,&
             1X,'     INPUT MATRIX B ( REAL , IMAGINARY )',/)
 6040 FORMAT(/,&
             1X,' **  OUTPUT  **',/,/,&
             1X,'     IERR =',I5,/)
 6050 FORMAT(/,&
             1X,14X,A11,22X,A11)
 6060 FORMAT(1X,12X,1PD14.7,19X,1PD14.7)
 6070 FORMAT(1X,5X,F12.8,' ,',F12.8,7X,F12.8,' ,',F12.8)
 6080 FORMAT(1X,' RE(A) = ',D10.2,' IM(A) = ',D10.2,&
                ' RE(B) = ',D10.2,' IM(B) = ',D10.2,&
                ' I,J = ',I2,3X,I2)
      END
```

(d) Output results

```
*** ZCGKAA ***

 ** INPUT **

    N =    4    LNA =    4   LNB =    4


    INPUT MATRIX A ( REAL , IMAGINARY )

    (  8.0,  0.0)(  3.0,  0.0)(  1.0, -2.0)( -1.0, -2.0)
    (  3.0,  0.0)(  9.0,  0.0)(  1.0,  2.0)( -1.0,  2.0)
    (  1.0,  2.0)(  1.0, -2.0)( 10.0,  0.0)( -3.0,  0.0)
    ( -1.0,  2.0)( -1.0, -2.0)( -3.0,  0.0)( 11.0,  0.0)

    INPUT MATRIX B ( REAL , IMAGINARY )

    (  8.0,  0.0)(  3.0,  0.0)(  1.0,  2.0)( -1.0,  2.0)
    (  3.0,  0.0)(  9.0,  0.0)(  1.0, -2.0)( -1.0, -2.0)
    (  1.0, -2.0)(  1.0,  2.0)( 10.0,  0.0)( -3.0,  0.0)
    ( -1.0, -2.0)( -1.0,  2.0)( -3.0,  0.0)( 11.0,  0.0)

 ** OUTPUT **

    IERR =    0


              EIGENVALUE                         EIGENVALUE
            1.6653903D+01                      3.6956492D+01

             EIGENVECTOR                        EIGENVECTOR
       -1.62644471 ,  0.00000000         -0.65982846 , -0.00000000
        1.40924217 , -0.00381901          0.61762116 , -0.09721828
        0.02962467 ,  0.54006351          0.11386209 ,  1.98647301
       -0.01716825 ,  0.50976217          0.09493235 ,  1.71080318

              EIGENVALUE                         EIGENVALUE
            1.0637116D+02                      2.1801844D+02

             EIGENVECTOR                        EIGENVECTOR
        1.75695717 ,  0.00000000          1.35366373 ,  0.00000000
        2.09207784 , -0.08597803          1.49511825 ,  0.05961668
       -1.02812336 ,  0.06200047          2.16426058 ,  0.05923384
        1.33921828 ,  0.00664669         -2.45129806 , -0.03970066
```

4.20.2 ZCGKAN, CCGKAN All Eigenvalues of Hermitian Matrices (Generalized Eigenvalue Problem $BAz = \lambda z$, B: Positive)

(1) **Function**

Generalized eigenvalue problem

$$BA\boldsymbol{z} = \lambda \boldsymbol{z}$$

(A: Hermitian, B: Positive Hermitian) is solved by using the Cholesky method, the Householder method and QR method to obtain all eigenvalues λ .

(2) **Usage**

Double precision:
CALL ZCGKAN (AR, AI, LNA, N, BR, BI, LNB, E, WORK, IERR)
Single precision:
CALL CCGKAN (AR, AI, LNA, N, BR, BI, LNB, E, WORK, IERR)

(3) **Arguments**

D:Double precision real Z:Double precision complex I: $\left\{ \begin{array}{l} \text{INTEGER(4) as for 32bit Integer} \\ \text{INTEGER(8) as for 64bit Integer} \end{array} \right\}$
R:Single precision real C:Single precision complex

No.	Argument	Type	Size	Input/ Output	Contents
1	AR	$\left\{\begin{matrix}\text{D}\\\text{R}\end{matrix}\right\}$	LNA, N	Input	Real part of Hermitian matrix A
				Output	Input-time contents are not retained.
2	AI	$\left\{\begin{matrix}\text{D}\\\text{R}\end{matrix}\right\}$	LNA, N	Input	Imaginary part of Hermitian matrix A
				Output	Input-time contents are not retained.
3	LNA	I	1	Input	Adjustable dimension of arrays AR and AI
4	N	I	1	Input	Order of matrices A and B
5	BR	$\left\{\begin{matrix}\text{D}\\\text{R}\end{matrix}\right\}$	LNB, N	Input	Real part of Hermitian matrix B
				Output	Input-time contents are not retained.
6	BI	$\left\{\begin{matrix}\text{D}\\\text{R}\end{matrix}\right\}$	LNB, N	Input	Imaginary part of Hermitian matrix B
				Output	Input-time contents are not retained.
7	LNB	I	1	Input	Adjustable dimension of arrays BR and BI
8	E	$\left\{\begin{matrix}\text{D}\\\text{R}\end{matrix}\right\}$	N	Output	Eigenvalues λ
9	WORK	$\left\{\begin{matrix}\text{D}\\\text{R}\end{matrix}\right\}$	4 × N	Work	Work area
10	IERR	I	1	Output	Error indicator

(4) **Restrictions**

(a) $1 \leq \text{N} \leq \text{LNA}, \text{LNB}$

(5) **Error indicator**

IERR value	Meaning	Processing
0	Normal termination.	
1000	N was equal to 1.	E(1) ← A(1, 1) × B(1, 1) is performed.
3000	Restriction (a) was not satisfied.	Processing is aborted.
4000	B was not positive definite.	
5000	The sequence did not converge in the step where the eigenvalue was obtained.	

(6) **Notes**

(a) Arrays AR, AI, BR and BI should be stored only in the upper triangular portions.

(b) Eigenvalues are stored in ascending order.

(c) 4.20.1 $\left\{\begin{matrix}\text{ZCGKAA}\\\text{CCGKAA}\end{matrix}\right\}$ should be used if the eigenvectors are needed.

(d) 4.19.2 $\left\{\begin{matrix}\text{ZCGJAN}\\\text{CCGJAN}\end{matrix}\right\}$ should be used if matrix A is only positive.

4.21 GENERALIZED EIGENVALUE PROBLEM FOR A REAL SYMMETRIC BAND MATRIX (SYMMETRIC BAND TYPE)

4.21.1 DCGBFF, RCGBFF Eigenvalues and Eigenvectors of a Real Symmetric Band Matrix (Generalized Eigenvalue Problem)

(1) **Function**

DCGBFF or RCGBFF uses the subspace method to obtain the eigenvalues having the m largest or m smallest absolute values of the real symmetric band matrix (symmetric band type) generalized eigenvalue problem $A\boldsymbol{x} = \lambda B\boldsymbol{x}$ (A: Real symmetric band matrix, B: Positive symmetric band matrix) and to obtain the corresponding eigenvectors.

(2) **Usage**

Double precision:

```
CALL DCGBFF (A, LMA, N, MAB, B, LMB, MBB, M, ITOL, NITE, E, VE, LNV, MST,
             IS1, IS2, W1, IW1, IERR)
```

Single precision:

```
CALL RCGBFF (A, LMA, N, MAB, B, LMB, MBB, M, ITOL, NITE, E, VE, LNV, MST,
             IS1, IS2, W1, IW1, IERR)
```

(3) **Arguments**

D:Double precision real Z:Double precision complex I: $\left\{\begin{array}{l}\text{INTEGER(4) as for 32bit Integer}\\\text{INTEGER(8) as for 64bit Integer}\end{array}\right\}$
R:Single precision real C:Single precision complex

No.	Argument	Type	Size	Input/ Output	Contents
1	A	$\left\{\begin{matrix}D\\R\end{matrix}\right\}$	LMA, N	Input	Real symmetric band matrix A (symmetric band type) (See Appendix B).
				Output	Input-time contents are not retained.
2	LMA	I	1	Input	Adjustable dimension of array A.
3	N	I	1	Input	Order of matrices A and B.
4	MAB	I	1	Input	Band width of matrix A.
5	B	$\left\{\begin{matrix}D\\R\end{matrix}\right\}$	LMB, N	Input	Positive symmetric band matrix B (symmetric band type).
6	LMB	I	1	Input	Adjustable dimension of array B.
7	MBB	I	1	Input	Band width of matrix B.
8	M	I	1	Input	The number of m of eigenvalues to be obtained.
9	ITOL	I	1	Input	Tolerance used for convergence test (See Note (b)).
10	NITE	I	1	Input	Maximum iteration count (See Note (d)).

No.	Argument	Type	Size	Input/ Output	Contents
11	E	$\left\{\begin{matrix}\mathrm{D}\\\mathrm{R}\end{matrix}\right\}$	See Contents	Output	Eigenvalues **Size**: $\min(2 \times \mathrm{M}, \mathrm{N}, \mathrm{M}+8)$
12	VE	$\left\{\begin{matrix}\mathrm{D}\\\mathrm{R}\end{matrix}\right\}$	See Contents	Output	Eigenvectors (column vector) corresponding to each eigenvalue. **Size**: $(\mathrm{LNV}, \min(2 \times \mathrm{M}, \mathrm{N}, \mathrm{M}+8))$
13	LNV	I	1	Input	Adjustable dimension of array VE.
14	MST	I	1	Output	Number of eigenvalues not calculated (See Note (e)).
15	IS1	I	1	Input	Processing switch. $\mathrm{IS1} \leq 0$: Obtain eigenvalues having the smallest absolute values. $\mathrm{IS1} > 0$: Obtain eigenvalues having the largest absolute values.
16	IS2	I	1	Input	Sturm sequence check switch. $\mathrm{IS2} \leq 0$: Do not check. $\mathrm{IS2} > 0$: Check.
17	W1	$\left\{\begin{matrix}\mathrm{D}\\\mathrm{R}\end{matrix}\right\}$	See Contents	Work	Work area **Size**: If $\mathrm{IS2} \leq 0$: $2 \times \mathrm{N} \times q + q \times q + 2 \times q + \mathrm{N}$ If $\mathrm{IS2} > 0$: $2 \times \mathrm{N} \times q + q \times q + 2 \times q + \mathrm{N} + \ell \times \mathrm{N}$ Here, $q = \min(2 \times \mathrm{M}, \mathrm{N}, \mathrm{M}+8)$. $\ell = \mathrm{MAB} + 1 \quad (\mathrm{IS1} \leq 0)$ $\ell = \mathrm{MBB} + 1 \quad (\mathrm{IS1} > 0)$
18	IW1	I	N	Work	Work area
19	IERR	I	1	Output	Error indicator

(4) **Restrictions**

(a) $0 < \mathrm{N} \leq \mathrm{LNV}$

(b) $0 \leq \mathrm{MAB} < \mathrm{N}$
$0 \leq \mathrm{MBB} < \mathrm{N}$

(c) $\mathrm{MAB} < \mathrm{LMA}$

(d) $\mathrm{MBB} < \mathrm{LMB}$

(e) $0 < \mathrm{M} \leq \mathrm{N}$

(5) **Error indicator**

IERR value	Meaning	Processing
0	Normal termination.	
1000	N was equal to 1.	$\mathrm{E}(1) \leftarrow \mathrm{A}(1,1)/\mathrm{B}(1,1)$ and $\mathrm{VE}(1,1) \leftarrow 1.0$ are performed.
3000	Restriction (a), (b), (c), (d) or (e) was not satisfied.	Processing is aborted.
4000	An error occurred during processing.	
$5000+i$	The sequence did not converge within the specified number of iteration.	Processing is aborted after obtaining up to the i-th eigenvalue and eigenvector.

(6) **Notes**

(a) This subroutine is effective when the number of eigenvalues to be obtained is very small (m≪N) relative to the order of the matrix. Otherwise, you should use other subroutines such 4.14.1 $\left\{\begin{matrix}\text{DCGSAA}\\\text{RCGSAA}\end{matrix}\right\}$, 4.13.1 $\left\{\begin{matrix}\text{DCGGAA}\\\text{RCGGAA}\end{matrix}\right\}$.

(b) This subroutine considers that the eigenvalue has converged if the following condition is satisfied. At this time, the eigenvector has a precision greater than or equal to ITOL/2.
$|\frac{a_i^n - a_i^{n-1}}{a_i^n}| \leq 10.0^{-\mathrm{ITOL}}$ (a_i^n: i-th eigenvalue after the n-th iteration)
If the input value of ITOL is less than or equal to 0 or greater than $-$ LOG10 (ε), then the optimum value is automatically set internally. (ε: Unit for determining error).

(c) Eigenvalues are stored in array E in ascending (or descending) order of their absolute values.

(d) If the input value of NITE is less than or equal to 0, then 20 is used as the default values.

(e) This subroutine has a function that checks whether the Sturm sequence property was used for the calculated eigenvalues. Although the number of calculated eigenvalues is computed, the number of calculations increases at this time on the order of $\mathrm{N} \times \mathrm{MB}^2$. For example, assume that three eigenvalues having the smallest absolute values are to be obtained for the eigenvalue problem having 6, 5, 3, 2 and 1 as eigenvalues. If 5, 2 and 1 are obtained as solution eigenvalues at this time, then 1 is returned to MST since the value 3 was not obtained as a solution. This function is effective only if all eigenvalues are positive.

(7) **Example**

(a) Problem

Obtain all eigenvalues of $A\boldsymbol{x} = \lambda B\boldsymbol{x}$ and their corresponding eigenvectors, where matrices A and B are as follows:

$$A = \begin{bmatrix} 611 & 196 & -192 & 407 & -8 & 0 & 0 & 0 \\ 196 & 899 & 113 & -192 & -71 & -43 & 0 & 0 \\ -192 & 113 & 899 & 196 & 61 & 49 & 8 & 0 \\ 407 & -192 & 196 & 611 & 8 & 44 & 59 & -23 \\ -8 & -71 & 61 & 8 & 411 & -599 & 208 & 208 \\ 0 & -43 & 49 & 44 & -599 & 411 & 208 & 208 \\ 0 & 0 & 8 & 59 & 208 & 208 & 99 & -911 \\ 0 & 0 & 0 & -23 & 208 & 208 & -911 & 99 \end{bmatrix}$$

$$B = \begin{bmatrix} 171 & 18 & 33 & -21 & -17 & 0 & 0 & 0 \\ 18 & 171 & -21 & 33 & 13 & -17 & 0 & 0 \\ 33 & -21 & 171 & 18 & 25 & -36 & -17 & 0 \\ -21 & 33 & 18 & 171 & -36 & 25 & 13 & -17 \\ -17 & 13 & 25 & -36 & 171 & 18 & 33 & -21 \\ 0 & -17 & -36 & 25 & 18 & 171 & -21 & 33 \\ 0 & 0 & -17 & 13 & 33 & -21 & 171 & 18 \\ 0 & 0 & 0 & -17 & -21 & 33 & 18 & 171 \end{bmatrix}$$

(b) Input data

Matrix A, LMA=11, N=8, MAB=4, matrix B, LMB=11, MBB=4, M=3 and LNV=11.

(c) Main program

```
      PROGRAM BCGBFF
! *** EXAMPLE OF DCGBFF ***
      IMPLICIT REAL(8)(A-H,O-Z)
      CHARACTER*80 FMT
      PARAMETER ( LMA=11, LMB=11, LNV=10, LN=10, LNQ=10 )
      PARAMETER ( LW=LNQ*LNQ+2*LNQ+LN*(2*LNQ+1+LN) )
      DIMENSION A(LMA,LN), B(LMB,LN), E(LN), VE(LNV,LNQ),&
                W1(LW), IW1(LN)
!
      READ(5,*) N, MAB, MBB, M
      DO 10 J=1, MAB+1
         READ(5,*) (A(J,I), I=MAB-J+2, N)
   10 CONTINUE
      DO 20 J=1, MBB+1
         READ(5,*) (B(J,I), I=MBB-J+2, N)
   20 CONTINUE
!
      WRITE(6,1000) N, MAB, MBB, M
      WRITE(6,1100) 'A'
      DO 30 J=1, MAB+1
         WRITE(FMT,1200) (MAB-J+1)*7+6, N-MAB+J-1
         WRITE(6,FMT) (A(J,I), I=MAB-J+2, N)
   30 CONTINUE
      WRITE(6,1100) 'B'
      DO 40 J=1, MBB+1
         WRITE(FMT,1200) (MBB-J+1)*7+6, N-MBB+J-1
         WRITE(6,FMT) (B(J,I), I=MBB-J+2, N)
   40 CONTINUE
!
      CALL DCGBFF(A,LMA,N,MAB,B,LMB,MBB,M,0,0,E,VE,LNV,MST,0,1,W1,IW1,&
                  IERR)
!
      WRITE(6,1300) IERR
!
      DO 60 K=1, M-3, 4
         WRITE(6,1400) ('EIGENVALUE ', I=1, 4)
         WRITE(6,1500) (E(I), I=K, K+3)
         WRITE(6,1400) ('EIGENVECTOR', I=1, 4)
         DO 50 J=1, N
            WRITE(6,1500) (VE(J,I), I=K, K+3)
   50    CONTINUE
   60 CONTINUE
      IF(MOD(M,4).NE.0) THEN
```

```
          WRITE(6,1400) ('EIGENVALUE ', I=M/4*4+1, M)
          WRITE(6,1500) (E(I), I=M/4*4+1, M)
          WRITE(6,1400) ('EIGENVECTOR', I=M/4*4+1, M)
          DO 70 J=1, N
             WRITE(6,1500) (VE(J,I), I=M/4*4+1, M)
    70    CONTINUE
       ENDIF
       WRITE(6,1600) MST
       STOP
!
  1000 FORMAT(' ',/,/,&
              ' ***  DCGBFF  ***',/,/,&
              '  **  INPUT  **',/,/,&
              '      N   = ', I2,/,/,&
              '      MAB = ', I2,/,/,&
              '      MBB = ', I2,/,/,&
              '      M   = ', I2)
  1100 FORMAT(' ',/,&
              '      INPUT MATRIX ',A1,/)
  1200 FORMAT('( ,', I3,'X,', I2,'(F7.1))')
  1300 FORMAT(' ',/,/,&
              '  **  OUTPUT  **',/,/,&
              '      IERR = ', I4)
  1400 FORMAT(' ',/,1X, 4(5X, A11, 2X))
  1500 FORMAT(' ', 4(2X, 1PD14.7, 2X))
  1600 FORMAT(' ',/,&
              '      MISSED EIGENVALUES = ', I2)
       END
```

(d) Output results

```
***  DCGBFF  ***

 **  INPUT  **

     N   =  8

     MAB =  4

     MBB =  4

     M   =  3

     INPUT MATRIX A

                                       -8.0  -43.0    8.0  -23.0
                               407.0  -71.0   49.0   59.0  208.0
                      -192.0 -192.0   61.0   44.0  208.0  208.0
               196.0  113.0  196.0    8.0 -599.0  208.0 -911.0
        611.0  899.0  899.0  611.0  411.0  411.0   99.0   99.0

     INPUT MATRIX B

                                      -17.0  -17.0  -17.0  -17.0
                              -21.0   13.0  -36.0   13.0  -21.0
                        33.0   33.0   25.0   25.0   33.0   33.0
                18.0  -21.0   18.0  -36.0   18.0  -21.0   18.0
        171.0  171.0  171.0  171.0  171.0  171.0  171.0  171.0


 **  OUTPUT  **

     IERR =    0

     EIGENVALUE        EIGENVALUE        EIGENVALUE
  -2.8694668D-02     1.1418645D-01     4.6073986D+00

     EIGENVECTOR       EIGENVECTOR       EIGENVECTOR
  -2.9518370D-02    -3.3280520D-02     1.3357085D-03
   1.8549154D-02     1.2860714D-02    -2.3833439D-02
  -1.8508500D-02    -1.2749259D-02    -1.4890226D-02
   2.7733942D-02     3.5436570D-02     5.2891703D-04
   3.2017538D-02    -3.0821630D-02    -3.4967437D-02
   3.0482006D-02    -3.2110386D-02     3.4727146D-02
   1.5596117D-02    -1.6287379D-02    -2.6667696D-02
   1.7726852D-02    -1.3617717D-02     2.5869490D-02

     MISSED EIGENVALUES =  1
```

Appendix A

GLOSSARY

(1) **Matrix**

An $m \times n$ matrix A is rectangular array of $m \times n$ elements $a_{i,j}$ $(i = 1, 2, \cdots, m;\ j = 1, 2, \cdots, n)$ as shown below.

$$\begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{bmatrix}$$

The element $a_{i,j}$ is called the (i, j)-th element of matrix A. The elements of a matrix are considered to be complex or real numbers. In particular, a matrix having complex numbers as its elements is called a complex matrix, and a matrix having real numbers as its elements is called a real matrix. Also, if $m = n$, the matrix A is called square matrix.

The matrix A is sometimes denotes as (a_{ij}). In this manual, $(a_{i,j})$ is used for distinguishing between the row subscript i and column subscript j as necessary.

(2) **(Number) vector**

1 x n matrix is called a row vector of size n, and an m x 1 matrix is called a column vector of size m. Unless it is specifically necessary to distinguish between them, both of these are simply called vectors. Mathematically, a vector is defined as a more abstract concept. The "vector" described here is called a number vector. For the definition of an abstract vector, see the explanation of "vector space."

(3) **Matrix product**

The matrix product $AB = (c_{i,l})$ of the two matrices $A = (a_{i,j})$ and $B = (b_{k,l})$ is defined as follows

$$c_{i,l} = \sum_j a_{i,j} \cdot b_{j,l}$$

only when the number of columns in matrix A is equal to the number of rows in matrix B.

(4) **Matrix-vector product**

If the matrix B in the matrix product AB is a column vector $\boldsymbol{x}$, then the product $A\boldsymbol{x}$ is called the matrix-vector product.

(5) **Transpose of matrix**

The matrix $A' = (a_{j,i})$ formed by interchanging the rows and columns in $m \times n$ matrix $A = (a_{i,j})$ $(i = 1, 2, \cdots, m;\ j = 1, 2, \cdots, n)$ is called the transpose of matrix A and is represented by A^T. The transpose may be also represented as ${}^tA$.

(6) **(Main) diagonal of a matrix**

The list of elements $a_{i,i}$ $(i = 1, 2, \cdots, n)$ in an $n \times n$ square matrix $A = (a_{i,j})$ $(i, j = 1, 2, \cdots, n)$ is called the (main) diagonal, and the elements are called the (main) diagonal elements. Also, a matrix having nonzero elements only on the diagonal is called a diagonal matrix.

(7) **Unit matrix**

An $n \times n$ matrix $A = (a_{i,j})$ $(i, j = 1, 2, \cdots, n)$ in which all the diagonal elements $a_{i,i}$ $(i = 1, 2, \cdots, n)$ are 1 and all the non-diagonal elements are 0 is called a unit matrix and is represented using the symbol E or I. This satisfies $AE = EA = A$ for any matrix A.

(8) **Inverse matrix**

For a square matrix A, if a square matrix B exist that satisfies $AB = BA = E$ (where E is the unit matrix), then the matrix B is called the inverse matrix of matrix A and is represented by the symbol A^{-1}.

(9) **General inverse matrix**

For an $m \times n$ matrix A, an $n \times m$ matrix X that satisfies the following relationships exists uniquely. This matrix X, which is called the (Moore-Penrose) general inverse matrix of matrix A, is represented by the symbol $A^{\dagger}$.

- $AXA \ = \ A$
- $XAX \ = \ X$
- $(AX)^T \ = \ AX$
- $(XA)^T \ = \ XA$

(10) **Lower triangle and upper triangle of a matrix**

The collection of elements $a_{i,j}$ $(i > j)$ in an $n \times n$ square matrix $A = (a_{i,j})$ $(i, j = 1, 2, \cdots, n)$ is called the lower triangle and the collection of elements $a_{i,j}$ $(i < j)$ is called the upper triangle. The diagonal may also be included in the definition of the upper and lower triangles. A matrix having nonzero elements only in the lower triangle that includes the diagonal is called a lower triangular matrix, and a matrix having nonzero elements only in the upper triangle that includes the diagonal is called an upper triangular matrix.

(11) **Conjugate transpose matrix**

The transpose of a matrix having the complex conjugates of the elements of a complex matrix A as elements is called conjugate transpose matrix and is represented by the symbol A^*. If the elements of a matrix are real numbers, then $A^* = A^T$.

(12) **Symmetric matrix**

A square matrix for which $A = A^T$ holds is called a symmetric matrix. In a symmetric matrix, $a_{i,j} = a_{j,i}$.

(13) **Hermitian matrix**

A square matrix for which $A = A^*$ holds is called a Hermitian matrix. In a Hermitian matrix, $a_{i,j}$ and $a_{j,i}$ are complex conjugates.

(14) **Unitary matrix**

The square matrix U for which $UU^* = I$ (I is the unit matrix) holds is called the unitary matrix.

(15) **Orthogonal matrix**

The real square matrix A for which $AA^T = I$ (I is the unit matrix) holds is called the orthogonal matrix.

(16) **Subdiagonal of a matrix**

The list of elements $a_{i,i+p}$ $(i = 1, 2, \cdots, n-p)$ in an $n \times n$ square matrix $A = (a_{i,j})$ $(i, j = 1, 2, \cdots, n)$ is called the p-th upper subdiagonal, and the list of elements $a_{i+q,i}$ $(i = 1, 2, \cdots, n-q)$ is called the q-th lower subdiagonal. The elements are called the p-th upper subdiagonal elements and q-th lower subdiagonal elements, respectively. Also, both of these collectively may be referred to simply as subdiagonal elements.

(17) **Band matrix**

A matrix having nonzero elements only on the main diagonal and in several upper and lower subdiagonals near the main diagonal in an $n \times n$ square matrix $A = (a_{i,j})$ $(i, j = 1, 2, \cdots, n)$ is called a band matrix. If the subdiagonals containing nonzero elements that are furthest from the diagonal are the u-th upper subdiagonal and l-th lower subdiagonal, the values u and l are called the upper bandwidth and lower bandwidth, respectively. if $u = l$, this is simply called the bandwidth.

(18) **Tridiagonal matrix**

A matrix in which the upper and lower bandwidths are both 1 is called a tridiagonal matrix.

(19) **Hessenberg matrix**

A matrix in which all lower triangle elements except the first lower subdiagonal are zero in an $n \times n$ square matrix $A = (a_{i,j})$ $(i, j = 1, 2, \cdots, n)$ is called a Hessenberg matrix. To obtain the eigenvalues of a matrix, a general matrix is converted to this kind of matrix.

(20) **Quasi-upper triangular matrix**

An $n \times n$ square matrix $A = (a_{i,j})$ $(i, j = 1, 2, \cdots, n)$ for which at least one of every two consecutive subdiagonal elements of the first lower subdiagonal is 0 and all lower triangular elements excluding the first lower subdiagonal are 0 is called a quasi-upper triangular matrix. This is a special case of a Hessenberg matrix.

(21) **Sparse matrix**

In general, a matrix in which the number of nonzero elements is relatively small compared to the total number of elements is called a sparse matrix. If the arrangement of the elements within a sparse matrix has some kind of regularity and an effective method of solving a problem is created by making practical use of this regularity, this matrix is called a regular sparse matrix. A sparse matrix that is not a regular sparse matrix is called an irregular sparse matrix. For example, a band matrix having a small bandwidth is a type of regular sparse matrix.

(22) **Regular and singular matrices**

If a square matrix A has an inverse matrix, the matrix A is said to be regular. A matrix that is not regular is said to be singular. The solutions of system of simultaneous linear equations having a regular matrix as coefficients are uniquely determined. However, since calculations are actually performed using a finite number of digits, the effects of rounding errors cannot be avoided, and the distinction between a regular and singular matrix becomes ambiguous. For example, solutions may apparently be obtained even when a system of simultaneous linear equations is solved numerically using a mathematically singular matrix. Therefore, when solving a system of simultaneous linear equations having a nearly singular matrix as coefficients, sufficient testing is required concerning the appropriateness of solutions that are apparently obtained.

(23) **LU decomposition**

To use a direct method to solve the system of simultaneous linear equations $A\boldsymbol{x} = \boldsymbol{b}$, first decompose the coefficient matrix A into the product $A = LU$ of the lower triangular matrix L and upper triangular matrix U. This decomposition is called an LU decomposition, If this kind of decomposition is performed, the solution $\boldsymbol{x}$ of the system of simultaneous linear equations is obtained by sequentially solving the following equations:

$$\begin{aligned} L\boldsymbol{y} &= \boldsymbol{b} \\ U\boldsymbol{x} &= \boldsymbol{y} \end{aligned}$$

Since the coefficient matrix of these two simultaneous linear equations is a triangular matrix, they can be easily solved by using forward-substitution and backward-substitution. If the matrix A is regular, for example, if the diagonal elements of matrix L are fixed at 1, the LU decomposition of the matrix A is uniquely determined. Also, when solving a system of simultaneous linear equations, since LU decomposition generally is performed while performing partial pivoting, if P is a row exchange matrix due to pivoting, triangular matrices L and U for which $PA = LU$ is satisfied are obtained, respectively.

(24) U^{T}DU **decomposition**
If the coefficient matrix of a system of simultaneous linear equations is a symmetric matrix, the relationship $L = U^T D$ holds between the lower triangular matrix L and upper triangular matrix U obtained by performing an LU decomposition without performing pivoting. Here, D is a diagonal matrix. Therefore, the system of simultaneous linear equations can be solved by explicitly obtaining only D and one of L and U. The decomposition that explicitly obtains U and D from coefficient matrix is called the U^{T}DU decomposition.

(25) U^{*}DU **decomposition**
If the coefficient matrix of a system of simultaneous linear equations is a Hermitian matrix, the relationship $L = U^* D$ holds between the lower triangular matrix L and upper triangular matrix U obtained by performing an LU decomposition without performing pivoting. Here, D is a diagonal matrix. Therefore, the system of simultaneous linear equations can be solved by explicitly obtaining only D and one of L and U. The decomposition that explicitly obtains U and D from coefficient matrix is called the U^{*}DU decomposition.

(26) **Positive definite**
If a real symmetric matrix or Hermitian matrix A satisfies $\boldsymbol{x}^* A \boldsymbol{x} > 0$ for an arbitrary vector $\boldsymbol{x}$ ($\boldsymbol{x} \neq \mathbf{0}$), it is said to be positive (definite). If it satisfies $\boldsymbol{x}^* A \boldsymbol{x} < 0$, it is said to be negative. The fact that the matrix A is a positive definite matrix is equivalent to the following two condition.

(a) All of the eigenvalues of matrix A are positive.

(b) All principal minors of matrix A are positive.

Although, mathematically, an LU decomposition can be performed for a positive definite matrix without performing pivoting, if pivoting is not actually performed, an LU decomposition may not be able to be performed numerically with stability.

(27) **Real eigenvalue**
The eigenvalue of a real square matrix are all real if and only if the matrix is a product of two real symmetric matrices. Also, the eigenvalue of a complex square matrix are all real if and only if the matrix is a product of two Hermitian matrices.

(28) **Diagonally dominant**
If the following holds for an $n \times n$ square matrix $A = (a_{i,j})$ $(i, j = 1, 2, \cdots, n)$

$$|a_{i,i}| \;>\; \sum_{\substack{j=1 \\ j\neq i}}^{n} |a_{i,j}| \;\; (i = 1, 2, \cdots, n)$$

matrix A is called a diagonally dominant matrix. Although, mathematically, an LU decomposition can be performed for a diagonally dominant matrix without performing pivoting, if pivoting is not actually performed, an LU decomposition may not be able to be performed numerically with stability.

(29) **Vector space**
If the set V satisfies conditions (a) and (b) V is called a vector space and its elements are called vectors.

(a) The sum $\boldsymbol{a}+\boldsymbol{b}$ of two elements $\boldsymbol{a}$ and $\boldsymbol{b}$ of V is uniquely determined as an element of V and satisfies the following properties.

   i. $(\boldsymbol{a}+\boldsymbol{b})+\boldsymbol{c}=\boldsymbol{a}+(\boldsymbol{b}+\boldsymbol{c})$ (associative law)
      Where, $\boldsymbol{a}$, $\boldsymbol{b}$ and $\boldsymbol{c}$ are arbitrary elements of V.

   ii. $\boldsymbol{a}+\boldsymbol{b}=\boldsymbol{b}+\boldsymbol{a}$ (commutative law)
      Where, $\boldsymbol{a}$ and $\boldsymbol{b}$ are arbitrary elements of V.

   iii. An element $\mathbf{0}$ of V, which is called the zero vector, exists and satisfies $\boldsymbol{a}+\mathbf{0}=\boldsymbol{a}$ for an arbitrary element $\boldsymbol{a}$ of V.

   iv. For an arbitrary element $\boldsymbol{a}$ of V, exactly one element $\boldsymbol{b}$ of V exists for which $\boldsymbol{a}+\boldsymbol{b}=\mathbf{0}$. This element $\boldsymbol{b}$ is represented as $-\boldsymbol{a}$.

(b) For an arbitrary element $\boldsymbol{a}$ of V and complex number c, $c\boldsymbol{a}$ (the c multiple of $\boldsymbol{a}$) is uniquely determined as an element of V and satisfies the following properties (scalar multiple).

   i. $c(\boldsymbol{a}+\boldsymbol{b})=c\boldsymbol{a}+c\boldsymbol{b}$ (vector distributive law)

   ii. $(c+d)\boldsymbol{a}=c\boldsymbol{a}+d\boldsymbol{a}$ (scalar distributive law)

   iii. $(cd)\boldsymbol{a}=c(d\boldsymbol{a})$

   iv. $1\boldsymbol{a}=\boldsymbol{a}$

(30) **Linear combination, linearly independent and linearly dependent**
The vector

$$c_1\boldsymbol{a}_1+\cdots+c_k\boldsymbol{a}_k$$

created from the k vectors $\boldsymbol{a}_1, \cdots, \boldsymbol{a}_k$ of vector space V and complex numbers $c_1, \cdots, c_k$ is called the linear combination of $\boldsymbol{a}_1, \cdots, \boldsymbol{a}_k$, and $c_1, \cdots, c_k$ are called its coefficients. For certain coefficients $c_1, \cdots, c_k$ that are not all zero, the set of vectors $\{\boldsymbol{a}_1, \cdots, \boldsymbol{a}_k\}$ is said to be linearly dependent if

$$c_1\boldsymbol{a}_1+\cdots+c_k\boldsymbol{a}_k=\mathbf{0}$$

and is said to be linearly independent otherwise.

(31) **Basis**
Let S be an arbitrary subset of vector space V, and let a collection of linearly independent vectors contained in S be $\{\boldsymbol{a}_1, \cdots, \boldsymbol{a}_k\}$. For an arbitrary vector $\boldsymbol{b}$ of S, if $\{\boldsymbol{a}_1, \cdots, \boldsymbol{a}_k, \boldsymbol{b}\}$ is linearly dependent, $\{\boldsymbol{a}_1, \cdots, \boldsymbol{a}_k\}$ is said to be the maximum set in S. When the vector space V itself is taken as S, this collection of linearly independent vectors is called the basis of vector space V. The number of vectors constituting the basis of V is called the dimension of V. Also, if we let an arbitrary basis of an n-dimensional vector space V_n be $\{\boldsymbol{u}_1, \cdots, \boldsymbol{u}_n\}$, then an arbitrary vector $\boldsymbol{a}$ of V_n is represented uniquely as a linear combination of $\{\boldsymbol{u}_1, \cdots, \boldsymbol{u}_n\}$.

(32) **(Vector) subspace**
A subset L of vector space V is called a (vector) subspace of V if the following conditions (a) and (b) are satisfied.

(a) If $\boldsymbol{a}, \boldsymbol{b} \in L$, then $\boldsymbol{a}+\boldsymbol{b} \in L$

(b) If $\boldsymbol{a} \in L$ and c is a complex number, $c\boldsymbol{a} \in L$

(33) **Linear transformation**

Let V_n and V_m be n-dimensional and m-dimensional vector spaces, respectively. If the mapping $\boldsymbol{A} : V_n \to V_m$ that associates each element $\boldsymbol{x}$ of V_n with an element $\boldsymbol{A}(\boldsymbol{x})$ of V_m satisfies the following two conditions, $\boldsymbol{A}$ is said to be a linear transformation from V_n to V_m.

(a) $\boldsymbol{A}(\boldsymbol{x}_1 + \boldsymbol{x}_2) = \boldsymbol{A}(\boldsymbol{x}_1) + \boldsymbol{A}(\boldsymbol{x}_1) \quad \boldsymbol{x}_1,\ \boldsymbol{x}_2 \in V_n$

(b) $\boldsymbol{A}(c\boldsymbol{x}) = c\boldsymbol{A}(\boldsymbol{x}) \quad \boldsymbol{x} \in V_n$ and c : a complex number

If we let a single basis of V_n and V_m, respectively, be $\{\boldsymbol{u}_1,\ \cdots,\ \boldsymbol{u}_n\}$ and $\{\boldsymbol{v}_1,\ \cdots,\ \boldsymbol{v}_m\}$, then $\boldsymbol{A}(\boldsymbol{x})$ is determined for an arbitrary $\boldsymbol{x} \in V_n$ according to the coefficient matrix $A = (a_{i,j})$ of

$$\boldsymbol{A}(\boldsymbol{u}_j) = \sum_{i=1}^{m} a_{i,j}\boldsymbol{v}_i \quad (j = 1, \cdots, n)$$

The matrix A is called the representation matrix of the linear transformation $\boldsymbol{A}$ related to this basis. Also, if $\boldsymbol{A}(\boldsymbol{x}) = \boldsymbol{x}$ for $\boldsymbol{x} \in V_n$, it defines the linear transformation $\boldsymbol{E} : V_n \to V_n$, which is called the identity transformation. The representation matrix of the identity transformation always is the unit matrix E regardless of how the basis is taken.

(34) **Eigenvalue and eigenvector**

For a linear transformation $\boldsymbol{A}$ within an n-dimensional vector space V_n, if there exists a number λ and a vector $\boldsymbol{x}$ ($\boldsymbol{x} \neq \boldsymbol{0}$) such that

$$\boldsymbol{A}(\boldsymbol{x}) = \lambda\boldsymbol{x}, \text{ that is, } (\boldsymbol{A} - \lambda\boldsymbol{E})(\boldsymbol{x}) = \boldsymbol{0}$$

is satisfied, then λ is called an eigenvalue of $\boldsymbol{A}$ and $\boldsymbol{x}$ is called the eigenvector belonging to the eigenvalue λ. Here, $\boldsymbol{E}$ is the identity transformation. If we fix a single basis within V_n, let the representation matrix of the linear transformation $\boldsymbol{A}$ be A, and let the number vector corresponding to the eigenvector $\boldsymbol{x}$ be $\hat{\boldsymbol{x}}$, then the eigenvalue λ and $\hat{\boldsymbol{x}}$ satisfy the following equation.

$$A\hat{\boldsymbol{x}} = \lambda\hat{\boldsymbol{x}}$$

Here, $\hat{\boldsymbol{x}}$ is represented using the components $x_1,\ \cdots,\ x_n$ of $\boldsymbol{x}$ as

$$\hat{\boldsymbol{x}} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$$

Normally, λ and $\hat{\boldsymbol{x}}$ are called the eigenvalue and eigenvector of matrix A, respectively. These terms are also used in this manual. Also, no distinction is made between the number vector and vector, which are represented as $\boldsymbol{x}$. Since the collection of all the vectors belonging to eigenvalue λ of the linear transformation $\boldsymbol{A} : V_n \to V_n$ together with the zero vector $\boldsymbol{0}$ form a single vector space, this is called the eigenvector space belonging to the eigenvalue λ of $\boldsymbol{A}$.

(35) **Invariant subspace**

For the linear transformation $\boldsymbol{A}$ within the vector space V_n, if the subspace U of V_n has the property

$$\boldsymbol{A}(U) \subseteq U$$

that is, if $\boldsymbol{A}\boldsymbol{x} \in U$ for an arbitrary vector $\boldsymbol{x}$, then U is said to be invariant relative to $\boldsymbol{A}$. In particular, the eigenvector space of $\boldsymbol{A}$ is invariant relative to $\boldsymbol{A}$. An invariant subvector space is called an invariant subspace.

(36) **Plane rotation**

The orthogonal transformation specified by the following kind of matrix $S_{k:l}(\theta)$ is called a plane rotation.

$$S_{kl}(\theta) = \begin{bmatrix} E_{1:k-1} & O_{1:k-1,k:l} & O_{1:k-1,l:n} \\ O_{k:l,1:k-1} & T_{k:l}(\theta) & O_{k:l,l:n} \\ O_{l:n,1:k-1} & O_{l:n,k:l} & E_{l:n} \end{bmatrix}$$

Here, $T_{k:l}(\theta)$ is defined as follows:

$$T_{k:l}(\theta) = \begin{bmatrix} \cos\theta & O_{k:k,k+1:l-1} & -\sin\theta \\ O_{k+1:l-1,k:k} & E_{k+1:l-1} & O_{k+1:l-1,l:l} \\ \sin\theta & O_{l:l,k+1:l-1} & \cos\theta \end{bmatrix}$$

$E_{p:q}$ is the $q-p+1$-dimensional unit matrix shown below:

$$E_{p:q} = \begin{array}{cl} \begin{bmatrix} 1 & & & 0 \\ & 1 & & \\ & & \ddots & \\ 0 & & & 1 \end{bmatrix} & \begin{array}{l} (p \\ (p+1 \\ \vdots \\ (q \end{array} \end{array}$$

and $O_{p:r,q:s}$ is the $r-p+1 \times s-q+1$-dimensional zero matrix shown below:

$$O_{p:r,q:s} = \begin{array}{cl} \begin{array}{cccc} \underbrace{q} & \underbrace{q+1} & \cdots & \underbrace{s} \end{array} & \\ \begin{bmatrix} 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix} & \begin{array}{l} (p \\ (p+1 \\ \vdots \\ (r \end{array} \end{array}$$

Now, if the submatrix $A_{p:r,q:s}$ of $A = (a_{i,j})$ $(i = 1, 2, \cdots, n; j = 1, 2, \cdots, n)$ is defined as follows:

$$A_{p:r,q:s} = \begin{bmatrix} a_{p,q} & a_{p,q+1} & \cdots & a_{p,s} \\ a_{p+1,q} & a_{p+1,q+1} & \cdots & a_{p+1,s} \\ \vdots & \vdots & \ddots & \vdots \\ a_{r,q} & a_{r,q+1} & \cdots & a_{r,s} \end{bmatrix}$$

the matrix A is represented as follows:

$$A = \begin{bmatrix} A_{1:k-1,1:k-1} & A_{1:k-1,k:l} & A_{1:k-1,l+1:n} \\ A_{k:l,1:k-1} & A_{k:l,k:l} & A_{k:l,l+1:n} \\ A_{l+1:n,1:k-1} & A_{l+1:n,k:l} & A_{l+1:n,l+1:n} \end{bmatrix}$$

At this time, since $S_{k:l}(\theta)A$ and $T_{k:l}(\theta)A_{k:l,q:s}$ are as follows:

$$S_{k:l}(\theta)A = \begin{bmatrix} A_{1:k-1,1:k-1} & A_{1:k-1,k:l} & A_{1:k-1,l+1:n} \\ T_{k:l}(\theta)A_{k:l,1:k-1} & T_{k:l}(\theta)A_{k:l,k:l} & T_{k:l}(\theta)A_{k:l,l+1:n} \\ A_{l+1:n,1:k-1} & A_{l+1:n,k:l} & A_{l+1:n,l+1:n} \end{bmatrix}$$

$$T_{k:l}(\theta)A_{k:l,q:s} = \begin{bmatrix} \cos\theta a_{k,q} - \sin\theta a_{l,q} & \cdots & \cos\theta a_{k,r} - \sin\theta a_{l,s} \\ a_{k+1,q} & \cdots & a_{k+1,r} \\ \vdots & \cdots & \vdots \\ a_{l-1,q} & \cdots & a_{l-1,r} \\ \sin\theta a_{k,q} + \cos\theta a_{l,q} & \cdots & \sin\theta a_{k,r} + \cos\theta a_{l,s} \end{bmatrix}$$

if θ is determined so that $\tan\theta = \frac{a_{l,i}}{a_{k,i}}$ or $\tan\theta = -\frac{a_{l,i}}{a_{k,i}}(i = q, \cdots, s)$ is satisfied, then an arbitrary element among the elements of column k and column l of $S_{k:l}(\theta)A$ can be set to zero. Now, since the following relationship holds:

$$AS_{k:l}(-\theta) = \begin{bmatrix} A_{1:k-1,1:k-1} & A_{1:k-1,k:l}T_{k:l}(-\theta) & A_{1:k-1,l+1:n} \\ A_{k:l,1:k-1} & A_{k:l,k:l}T_{k:l}(-\theta) & A_{k:l,l+1:n} \\ A_{l+1:n,1:k-1} & A_{l+1:n,k:l}T_{k:l}(-\theta) & A_{l+1:n,l+1:n} \end{bmatrix}$$

$$A_{p:r,k:l}T_{k:l}(-\theta) = \begin{bmatrix} \cos\theta a_{p,k} - \sin\theta a_{p,l} & a_{p,k+1} & \cdots & a_{p,l-1} & \sin\theta a_{p,k} + \cos\theta a_{p,l} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \cos\theta a_{r,k} - \sin\theta a_{r,l} & a_{r,k+1} & \cdots & a_{r,l-1} & \sin\theta a_{r,k} + \cos\theta a_{r,l} \end{bmatrix}$$

if θ is determined so that $\tan\theta = \frac{a_{i,l}}{a_{i,k}}$ or $\tan\theta = -\frac{a_{i,l}}{a_{i,k}}(i = p, \cdots, r)$ is satisfied, then an arbitrary element among the elements of column k and column l of $AS_{k:l}(-\theta)$ can be set to zero. Now, since the following relationship holds:

$$S_{k:l}(-\theta) = S_{k:l}(\theta)^T$$

and since $\tilde{A} = S_{k:l}(\theta)AS_{k:l}(-\theta)$ is as follows:

$$\tilde{A} = S_{k:l}(\theta)AS_{k:l}(-\theta) = \begin{bmatrix} A_{1:k-1,1:k-1} & A_{1:k-1,k:l}T_{k:l}(-\theta) & A_{1:k-1,l+1:n} \\ T_{k:l}(\theta)A_{k:l,1:k-1} & T_{k:l}(\theta)A_{k:l,k:l}T_{k:l}(-\theta) & T_{k:l}(\theta)A_{k:l,l+1:n} \\ A_{l+1:n,1:k-1} & A_{l+1:n,k:l}T_{k:l}(-\theta) & A_{l+1:n,l+1:n} \end{bmatrix}$$

if matrix A is a symmetric matrix, then by adjusting θ, either:

$$\tilde{a}_{k,j} = \tilde{a}_{j,k} = 0$$

or

$$\tilde{a}_{l,j} = \tilde{a}_{j,l} = 0$$

can be set for some $j(j \neq k, j \neq l)$, where the elements of $\tilde{A} = S_{k:l}(\theta)AS_{k:l}(-\theta)$ are represented by $(\tilde{a}_{i,j})$.

Appendix B

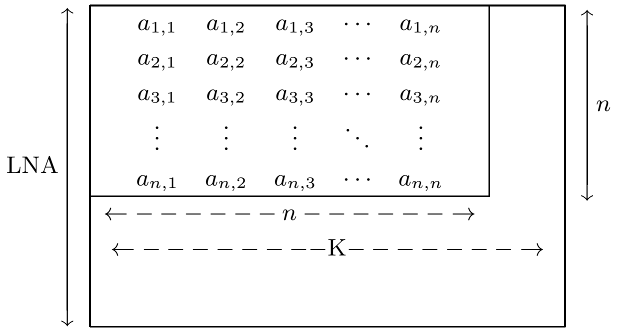
METHODS OF HANDLING ARRAY DATA

B.1 Methods of handling array data corresponding to matrix

Since the ASL subroutine library uses array data corresponding to matrix, this section describes various methods of handling arrays.

To call a subroutine 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.

Matrix Storage Mode Within an Array A


$a_{1,1}$ $a_{1,2}$ $a_{1,3}$ $\cdots$ $a_{1,n}$
$a_{2,1}$ $a_{2,2}$ $a_{2,3}$ $\cdots$ $a_{2,n}$
$a_{3,1}$ $a_{3,2}$ $a_{3,3}$ $\cdots$ $a_{3,n}$
$a_{n,1}$ $a_{n,2}$ $a_{n,3}$ $\cdots$ $a_{n,n}$
LNA
n
n
K

Remarks

a. LNA $\geq n$ and K $\geq n$ must hold.

b. Matrix element $a_{i,j}$ corresponds to the array element A(i, j).

Figure B−1 Matrix Storage Mode Within an Array A()

LNA is called an adjustable dimension. If a two-dimensional array is used as an argument, the adjustable must be passed to the subroutine as an argument in addition to the array name and order of the array. The matrix elements $a_{i,j}$ $(i = 1, 2, \cdots,$LNA$; j = 1, 2, \cdots,$K$)$ must correspond to the array element A(i, j) (i $= 1, 2, \cdots,$ LNA; j $= 1, 2, \cdots,$ K) , as follows on the main memory.

$$\begin{array}{ccccccc}
a_{1,1} & a_{2,1} & \cdots & a_{\mathrm{LNA},1} & a_{1,2} & a_{2,2} & \cdots \\
\updownarrow & \updownarrow & \cdots & \updownarrow & \updownarrow & \updownarrow & \cdots \\
\mathrm{A}(1,1) & \mathrm{A}(2,1) & \cdots & \mathrm{A}(\mathrm{LNA},1) & \mathrm{A}(1,2) & \mathrm{A}(2,2) & \cdots
\end{array}$$

Example DAM1AD (Real matrix addition)

Add 3×2 matrices A and B placing the sum in matrix C. If you declare arrays of size (5, 4), the declaration and CALL statements are as follows.

```
      REAL(8) A(5, 4), B(5, 4), C(5, 4)
      INTEGER IERR
C
      CALL DAM1AD(A, 5, 3, 2, B, 5, C, 5, IERR)
```

Data is stored in A as follows. Data are stored in B and C in the same way.

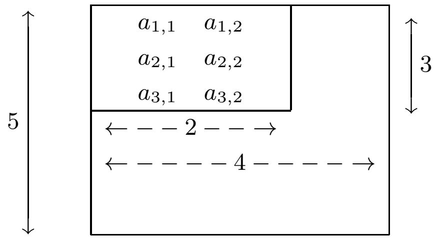


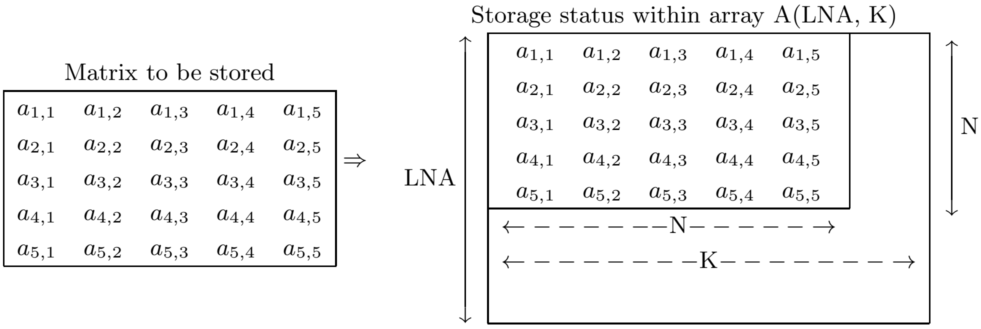
Figure B−2 Matrix Storage Mode Within an Array A

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.

B.2 Data storage modes

Matrix data storage modes differ according to the matrix type. Storage modes for each type of matrix are shown below.

B.2.1 Real matrix (two-dimensional array type)


Matrix to be stored
$a_{1,1}$ $a_{1,2}$ $a_{1,3}$ $a_{1,4}$ $a_{1,5}$
$a_{2,1}$ $a_{2,2}$ $a_{2,3}$ $a_{2,4}$ $a_{2,5}$
$a_{3,1}$ $a_{3,2}$ $a_{3,3}$ $a_{3,4}$ $a_{3,5}$
$a_{4,1}$ $a_{4,2}$ $a_{4,3}$ $a_{4,4}$ $a_{4,5}$
$a_{5,1}$ $a_{5,2}$ $a_{5,3}$ $a_{5,4}$ $a_{5,5}$
Storage status within array A(LNA, K)
LNA
N
N
K

Remarks

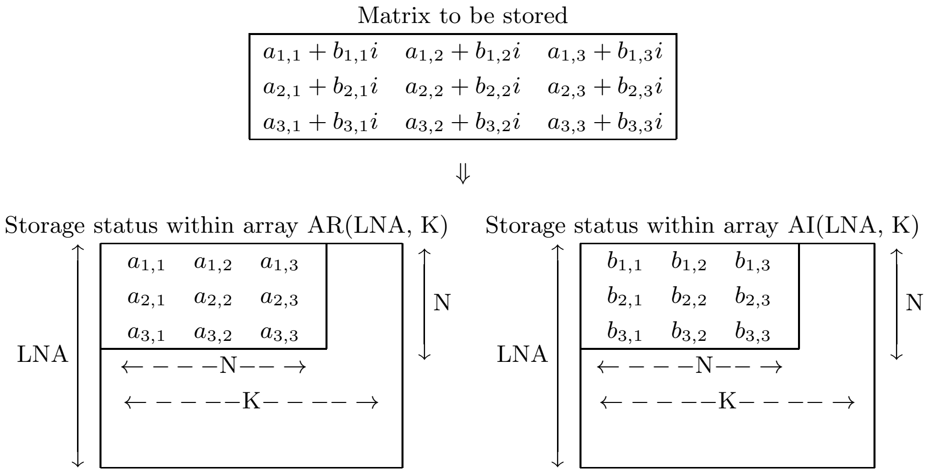
a. LNA ≥ N and K ≥ N must hold.

Figure B−3 Real Matrix (Two-Dimensional Array Type) Storage Mode

B.2.2 Complex matrix

(1) **Two-dimensional array type, real argument type**

Real and imaginary parts are stored in separate arrays.



Matrix to be stored

$$\begin{array}{|ccc|}\hline a_{1,1}+b_{1,1}i & a_{1,2}+b_{1,2}i & a_{1,3}+b_{1,3}i \\ a_{2,1}+b_{2,1}i & a_{2,2}+b_{2,2}i & a_{2,3}+b_{2,3}i \\ a_{3,1}+b_{3,1}i & a_{3,2}+b_{3,2}i & a_{3,3}+b_{3,3}i \\ \hline\end{array}$$

$\Downarrow$

Storage status within array AR(LNA, K)

$$\begin{array}{|ccc|}\hline a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,1} & a_{3,2} & a_{3,3} \\ \hline\end{array}$$

Storage status within array AI(LNA, K)

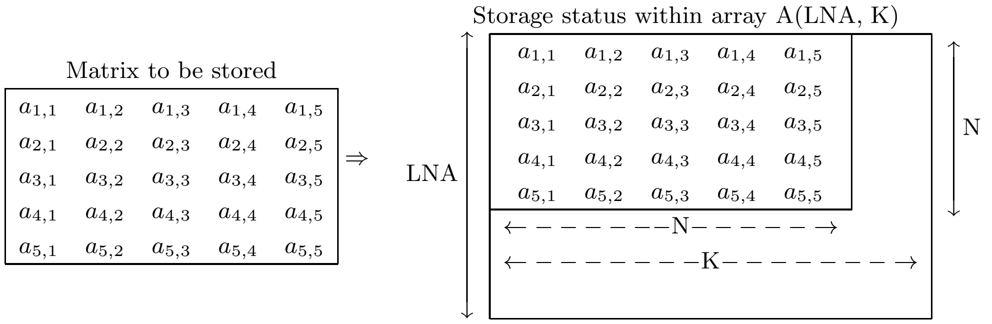
$$\begin{array}{|ccc|}\hline b_{1,1} & b_{1,2} & b_{1,3} \\ b_{2,1} & b_{2,2} & b_{2,3} \\ b_{3,1} & b_{3,2} & b_{3,3} \\ \hline\end{array}$$

Remarks

a. LNA ≥ N and K ≥ N must hold.

Figure B−4 Complex Matrix (Two-dimensional Array Type) (Real Argument Type) Storage Mode

(2) **Two-dimensional array type, complex argument type**



Matrix to be stored

$$\begin{array}{|ccccc|}\hline a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} & a_{1,5} \\ a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} & a_{2,5} \\ a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} & a_{3,5} \\ a_{4,1} & a_{4,2} & a_{4,3} & a_{4,4} & a_{4,5} \\ a_{5,1} & a_{5,2} & a_{5,3} & a_{5,4} & a_{5,5} \\ \hline\end{array} \Rightarrow$$

Storage status within array A(LNA, K)

$$\begin{array}{|ccccc|}\hline a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} & a_{1,5} \\ a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} & a_{2,5} \\ a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} & a_{3,5} \\ a_{4,1} & a_{4,2} & a_{4,3} & a_{4,4} & a_{4,5} \\ a_{5,1} & a_{5,2} & a_{5,3} & a_{5,4} & a_{5,5} \\ \hline\end{array}$$

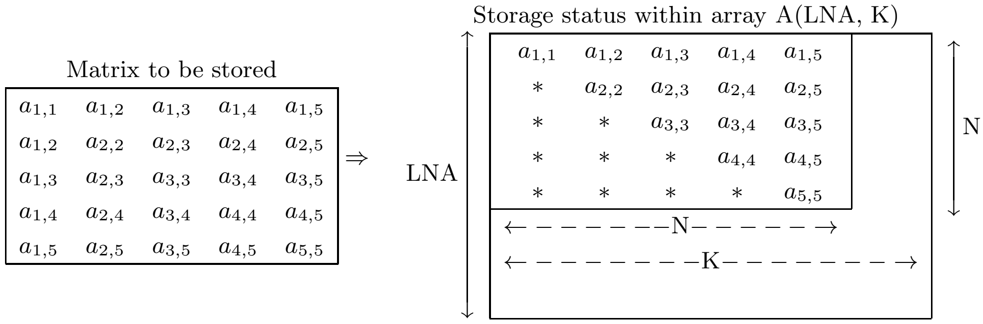
Remarks

a. LNA ≥ N and K ≥ N must hold.

Figure B−5 Complex Matrix (Two-dimensional Array Type)(Complex Argument Type) Storage Mode

B.2.3 Real symmetric matrix and positive symmetric matrix

(1) **Two-dimensional array type, upper triangular type**



Matrix to be stored

$$\begin{array}{ccccc} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} & a_{1,5} \\ a_{1,2} & a_{2,2} & a_{2,3} & a_{2,4} & a_{2,5} \\ a_{1,3} & a_{2,3} & a_{3,3} & a_{3,4} & a_{3,5} \\ a_{1,4} & a_{2,4} & a_{3,4} & a_{4,4} & a_{4,5} \\ a_{1,5} & a_{2,5} & a_{3,5} & a_{4,5} & a_{5,5} \end{array}$$

Storage status within array A(LNA, K)

$$\begin{array}{ccccc} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} & a_{1,5} \\ * & a_{2,2} & a_{2,3} & a_{2,4} & a_{2,5} \\ * & * & a_{3,3} & a_{3,4} & a_{3,5} \\ * & * & * & a_{4,4} & a_{4,5} \\ * & * & * & * & a_{5,5} \end{array}$$

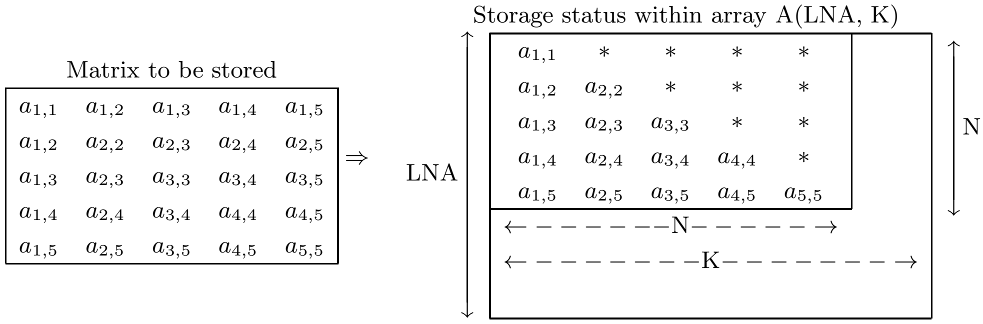
Remarks

a. The asterisk ($*$) indicates an arbitrary value.

b. LNA $\geq$ N and K $\geq$ N must hold.

Figure B−6 Real Symmetric Matrix (Two-dimensional Array Type) (Upper Triangular Type) Storage mode

(2) **Two-dimensional array type, lower triangular type**



Matrix to be stored

$$\begin{array}{ccccc} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} & a_{1,5} \\ a_{1,2} & a_{2,2} & a_{2,3} & a_{2,4} & a_{2,5} \\ a_{1,3} & a_{2,3} & a_{3,3} & a_{3,4} & a_{3,5} \\ a_{1,4} & a_{2,4} & a_{3,4} & a_{4,4} & a_{4,5} \\ a_{1,5} & a_{2,5} & a_{3,5} & a_{4,5} & a_{5,5} \end{array}$$

Storage status within array A(LNA, K)

$$\begin{array}{ccccc} a_{1,1} & * & * & * & * \\ a_{1,2} & a_{2,2} & * & * & * \\ a_{1,3} & a_{2,3} & a_{3,3} & * & * \\ a_{1,4} & a_{2,4} & a_{3,4} & a_{4,4} & * \\ a_{1,5} & a_{2,5} & a_{3,5} & a_{4,5} & a_{5,5} \end{array}$$

Remarks

a. The asterisk ($*$) indicates an arbitrary value.

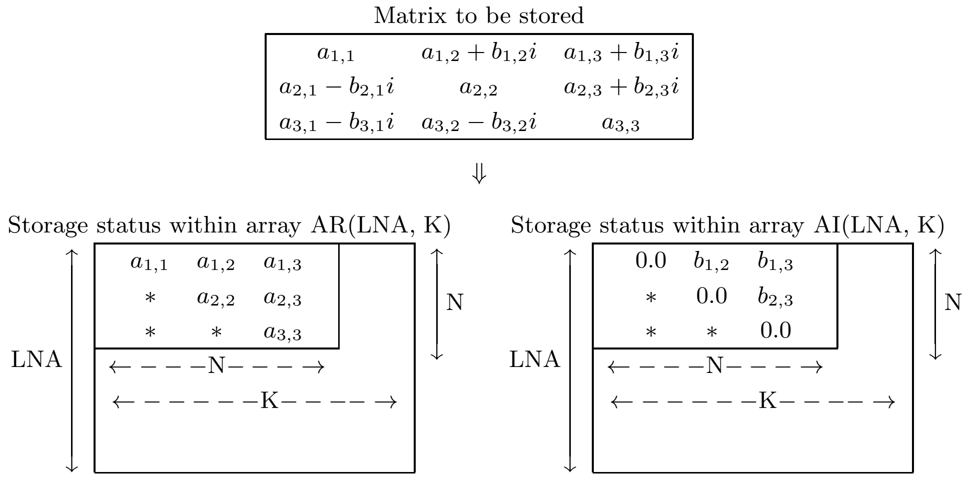
b. LNA $\geq$ N and K $\geq$ N must hold.

Figure B−7 Real Symmetric Matrix (Two-dimensional Array Type, Lower Triangular Type) Storage mode

B.2.4 Hermitian matrix

(1) **Two-dimensional array type, real argument type, upper triangular type**

Upper triangular portions of the real and imaginary parts are stored in separate arrays.



Remarks

a. The asterisk ($*$) indicates an arbitrary value.

b. LNA $\geq$ N and K $\geq$ N must hold.

Figure B−8 Hermitian Matrix (Two-dimensional Array Type) (Real Argument Type) (Upper Triangular Type) Storage Mode

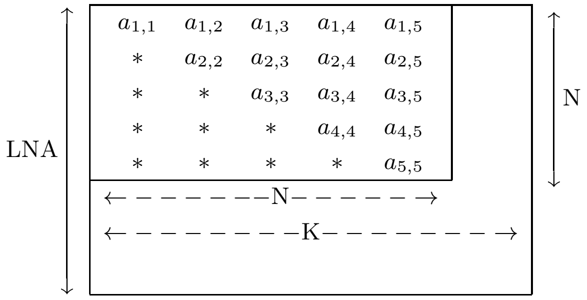
(2) Two-dimensional array type, complex argument type, upper triangular type

Matrix to be stored

$$\begin{array}{|ccccc|}\hline a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} & a_{1,5} \\ \overline{a_{1,2}} & a_{2,2} & a_{2,3} & a_{2,4} & a_{2,5} \\ \overline{a_{1,3}} & \overline{a_{2,3}} & a_{3,3} & a_{3,4} & a_{3,5} \\ \overline{a_{1,4}} & \overline{a_{2,4}} & \overline{a_{3,4}} & a_{4,4} & a_{4,5} \\ \overline{a_{1,5}} & \overline{a_{2,5}} & \overline{a_{3,5}} & \overline{a_{4,5}} & a_{5,5} \\ \hline \end{array}$$

⇓

Storage status within array A(LNA, K)



$$\begin{array}{ccccc} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} & a_{1,5} \\ * & a_{2,2} & a_{2,3} & a_{2,4} & a_{2,5} \\ * & * & a_{3,3} & a_{3,4} & a_{3,5} \\ * & * & * & a_{4,4} & a_{4,5} \\ * & * & * & * & a_{5,5} \end{array}$$

Remarks

a. The $\overline{x}$ indicates the complex conjugate of x.

b. The asterisk $*$ indicates an arbitrary value.

c. LNA $\geq$ N and K $\geq$ N must hold.

Figure B−9 Hermitian Matrix (Two-dimensional Array Type) (Complex Argument Type) (Upper Triangular Type) Storage Mode

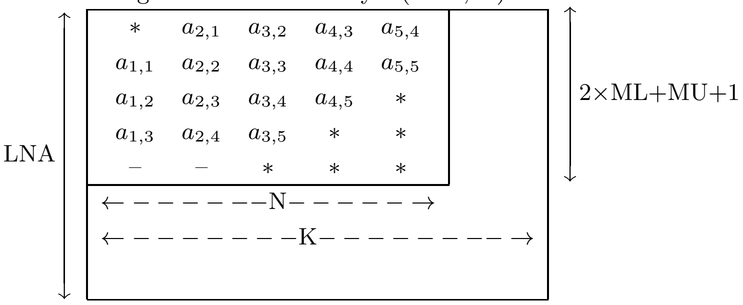
B.2.5 Real band matrix

Matrix to be stored

$$\begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & & 0 \\ a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} & \\ & a_{3,2} & a_{3,3} & a_{3,4} & a_{3,5} \\ & & a_{4,3} & a_{4,4} & a_{4,5} \\ 0 & & & a_{5,4} & a_{5,5} \end{bmatrix}$$

$$\Downarrow$$

Storage status within array A(LNA, K)



$$\begin{array}{ccccc} * & a_{2,1} & a_{3,2} & a_{4,3} & a_{5,4} \\ a_{1,1} & a_{2,2} & a_{3,3} & a_{4,4} & a_{5,5} \\ a_{1,2} & a_{2,3} & a_{3,4} & a_{4,5} & * \\ a_{1,3} & a_{2,4} & a_{3,5} & * & * \\ - & - & * & * & * \end{array}$$

Remarks

a. The asterisk $*$ indicates an arbitrary value.

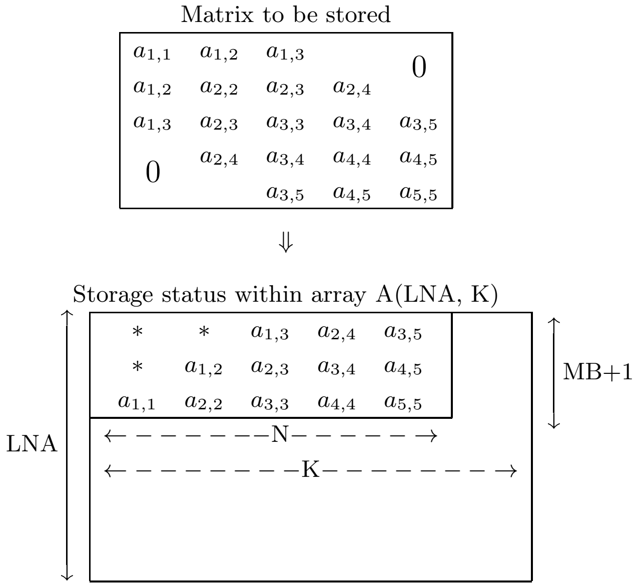
b. The area indicated by dashes (–) is required for an LU decomposition of the matrix.

c. MU is the upper band width and ML is the lower band width.

d. LNA $\geq 2 \times \mathrm{ML} + \mathrm{MU} + 1$ and K $\geq$ N must hold. (However, if no LU decomposition is to be performed, LNA $\geq \mathrm{ML} + \mathrm{MU} + 1$ and K $\geq$ N is sufficient.)

Figure B−10 Real Band Matrix (Band Type) Storage Mode

B.2.6 Real symmetric band matrix and positive symmetric matrix (symmetric band type)



Matrix to be stored

$$\begin{array}{|ccccc|}\hline a_{1,1} & a_{1,2} & a_{1,3} & & 0 \\ a_{1,2} & a_{2,2} & a_{2,3} & a_{2,4} & \\ a_{1,3} & a_{2,3} & a_{3,3} & a_{3,4} & a_{3,5} \\ 0 & a_{2,4} & a_{3,4} & a_{4,4} & a_{4,5} \\ & & a_{3,5} & a_{4,5} & a_{5,5} \\ \hline \end{array}$$

⇓

Storage status within array A(LNA, K)

$$\begin{array}{|ccccc|}\hline * & * & a_{1,3} & a_{2,4} & a_{3,5} \\ * & a_{1,2} & a_{2,3} & a_{3,4} & a_{4,5} \\ a_{1,1} & a_{2,2} & a_{3,3} & a_{4,4} & a_{5,5} \\ \hline \end{array}$$

Remarks

a. The asterisk $*$ indicates an arbitrary value.

b. MB is the band width.

c. LNA $\geq$ MB + 1 and K $\geq$ N must hold.

Figure B−11 Real Symmetric Band Matrix (Symmetric Band Type) Storage Mode

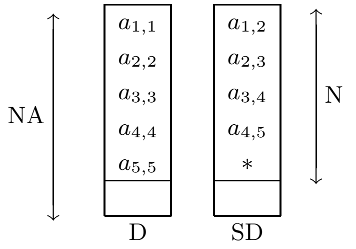
B.2.7 Real symmetric tridiagonal matrix and positive symmetric tridiagonal matrix (vector type)

Matrix to be stored

$$\begin{array}{|ccccc|}\hline a_{1,1} & a_{1,2} & & & 0 \\ a_{1,2} & a_{2,2} & a_{2,3} & & \\ & a_{2,3} & a_{3,3} & a_{3,4} & \\ 0 & & a_{3,4} & a_{4,4} & a_{4,5} \\ & & & a_{4,5} & a_{5,5} \\ \hline \end{array}$$

⇓

Storage status within arrays D(NA) (diagonal component) and SD(NA) (subdiagonal component)



Remarks

a. The asterisk $*$ indicates an arbitrary value.

b. NA $\geq$ N must hold.

Figure B−12 Real Symmetric Tridiagonal Matrix (Vector Type) Storage Mode

B.2.8 Triangular matrix

(1) Two-dimensional array type

The storage mode is the same as for a real symmetric matrix (two-dimensional array type) (upper triangular type) or a real symmetric matrix (two-dimensional array type) (lower triangular type).

B.2.9 Random sparse matrix (For symmetric matrix only)

(1) One-dimensional row-oriented list format (Symmetric case)

Matrix A to be stored

$$\begin{array}{|ccccccc|}\hline
a_{1,1} & a_{1,2} & 0.0 & a_{1,4} & 0.0 & a_{1,6} & 0.0 \\
a_{1,2} & a_{2,2} & a_{2,3} & a_{2,4} & 0.0 & 0.0 & a_{2,7} \\
0.0 & a_{2,3} & a_{3,3} & 0.0 & 0.0 & a_{3,6} & 0.0 \\
a_{1,4} & a_{2,4} & 0.0 & a_{4,4} & a_{4,5} & a_{4,6} & 0.0 \\
0.0 & 0.0 & 0.0 & a_{4,5} & a_{5,5} & 0.0 & 0.0 \\
a_{1,6} & 0.0 & a_{3,6} & a_{4,6} & 0.0 & a_{6,6} & 0.0 \\
0.0 & a_{2,7} & 0.0 & 0.0 & 0.0 & 0.0 & a_{7,7} \\ \hline
\end{array}$$

$\Downarrow$

Storage status of arrays A(NA), JA(NA) and IA(N+1)

	A	JA	IA	
	$a_{1,1}$	1	1	
	$a_{1,2}$	2	5	
	$a_{1,4}$	4	9	
	$a_{1,6}$	6	11	N+1
	$a_{2,2}$	2	14	
	$a_{2,3}$	3	15	
NA	$a_{2,4}$	4	16	
	$a_{2,7}$	7	17	
	$a_{3,3}$	3		
	$a_{3,6}$	6		
	$\vdots$	$\vdots$		
	$a_{7,7}$	7		

Remarks

a. N is the order of matrix A.

b. NA is the number of nonzero upper triangular elements.

c. A contains the nonzero upper triangular elements of matrix A, stored sequentially beginning with the first row.

d. JA contains the column numbers in the original matrix A of the elements stored in A.

e. IA contains values equal to the positions in array A of the diagonal elements. The N+1-th element contains the value NA+1.

Figure B−13 Storage of Symmetric Random Sparse Matrix (One-dimensional Row-oriented List Format)

(2) **One-dimensional column-oriented list format (Symmetric case)**

Matrix A to be stored

$$\begin{array}{|ccccccc|}
\hline
a_{1,1} & a_{1,2} & 0.0 & a_{1,4} & 0.0 & a_{1,6} & 0.0 \\
a_{1,2} & a_{2,2} & a_{2,3} & a_{2,4} & 0.0 & 0.0 & a_{2,7} \\
0.0 & a_{2,3} & a_{3,3} & 0.0 & 0.0 & a_{3,6} & 0.0 \\
a_{1,4} & a_{2,4} & 0.0 & a_{4,4} & a_{4,5} & a_{4,6} & 0.0 \\
0.0 & 0.0 & 0.0 & a_{4,5} & a_{5,5} & 0.0 & 0.0 \\
a_{1,6} & 0.0 & a_{3,6} & a_{4,6} & 0.0 & a_{6,6} & 0.0 \\
0.0 & a_{2,7} & 0.0 & 0.0 & 0.0 & 0.0 & a_{7,7} \\
\hline
\end{array}$$

$\Downarrow$

Storage status of arrays VALUES(NNZ), IRWIND(NNZ) and IPONTR(N+1)

	VALUES	IRWIND	IPONTR	
NNZ	$a_{1,1}$	1	1	N+1
	$a_{1,2}$	1	2	
	$a_{2,2}$	2	4	
	$a_{2,3}$	2	6	
	$a_{3,3}$	3	9	
	$a_{1,4}$	1	11	
	$a_{2,4}$	2	15	
	$a_{4,4}$	4	17	
	$a_{4,5}$	4		
	$a_{5,5}$	5		
	$a_{1,6}$	1		
	$a_{3,6}$	3		
	$a_{4,6}$	4		
	$a_{6,6}$	6		
	$a_{2,7}$	2		
	$a_{7,7}$	7		

ITYPE = $\boxed{2}$

Remarks

a. N is the order of Matrix A.

b. NNZ is the number of nonzero upper triangular elements.

c. VALUES contains the nonzero upper triangular elements of Matrix A, stored sequentially beginning with the first column.

d. IRWIND contains the row indices in the original matrix A of the elements stored in VALUES.

e. IPONTR contains values equal to the positions in array VALUES of the diagonal elements. Here, IPONTR(1) contains the value 1 and IPONTR(N+1) contains the value NNZ+1.

Figure B−14 Storage of Symmetric Random Sparse Matrix (One-dimensional Column-oriented List Format) for Upper Triangular Part

Matrix A to be stored

$$\begin{array}{|ccccccc|}
\hline
a_{1,1} & a_{2,1} & 0.0 & a_{4,1} & 0.0 & a_{6,1} & 0.0 \\
a_{2,1} & a_{2,2} & a_{3,2} & a_{4,2} & 0.0 & 0.0 & a_{7,2} \\
0.0 & a_{3,2} & a_{3,3} & 0.0 & 0.0 & a_{6,3} & 0.0 \\
a_{4,1} & a_{4,2} & 0.0 & a_{4,4} & a_{5,4} & a_{6,4} & 0.0 \\
0.0 & 0.0 & 0.0 & a_{5,4} & a_{5,5} & 0.0 & 0.0 \\
a_{6,1} & 0.0 & a_{6,3} & a_{6,4} & 0.0 & a_{6,6} & 0.0 \\
0.0 & a_{7,2} & 0.0 & 0.0 & 0.0 & 0.0 & a_{7,7} \\
\hline
\end{array}$$

$\Downarrow$

Storage status of arrays VALUES(NNZ), IRWIND(NNZ) and IPONTR(N+1)

	VALUES	IRWIND	IPONTR	
	$a_{1,1}$	1	1	
	$a_{2,1}$	2	5	
	$a_{4,1}$	4	9	
	$a_{6,1}$	6	11	
	$a_{2,2}$	2	14	N+1
	$a_{3,2}$	3	15	
	$a_{4,2}$	4	16	
	$a_{7,2}$	7	17	
NNZ	$a_{3,3}$	3		
	$a_{6,3}$	6		
	$a_{4,4}$	4		
	$a_{5,4}$	5		
	$a_{6,4}$	6		
	$a_{5,5}$	5		
	$a_{6,6}$	6		
	$a_{7,7}$	7		

ITYPE = $\boxed{2}$

Remarks

a. N is the order of Matrix A.

b. NNZ is the number of nonzero lower triangular elements.

c. VALUES contains the nonzero lower triangular elements of Matrix A, stored sequentially beginning with the first column.

d. IRWIND contains the row indices in the original matrix A of the elements stored in VALUES.

e. IPONTR contains values equal to the positions in array VALUES of the diagonal elements. IPONTR(1) contains the value 1 and IPONTR(N+1) contains the value NNZ+1.

Figure B−15 Storage of Symmetric Random Sparse Matrix (One-dimensional Column-oriented List Format) for Lower Triangular Part

B.2.10 Random sparse matrix

(1) ELLPACK format

Matrix A to be stored

$$\begin{array}{|ccccccc|}
\hline
a_{1,1} & a_{1,2} & 0.0 & a_{1,4} & 0.0 & a_{1,6} & 0.0 \\
a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} & 0.0 & 0.0 & a_{2,7} \\
0.0 & a_{3,2} & a_{3,3} & 0.0 & 0.0 & a_{3,6} & 0.0 \\
a_{4,1} & a_{4,2} & 0.0 & a_{4,4} & a_{4,5} & a_{4,6} & 0.0 \\
0.0 & 0.0 & 0.0 & a_{5,4} & a_{5,5} & 0.0 & 0.0 \\
a_{6,1} & 0.0 & a_{6,3} & a_{6,4} & 0.0 & a_{6,6} & a_{6,7} \\
0.0 & a_{7,2} & 0.0 & 0.0 & 0.0 & 0.0 & a_{7,7} \\
\hline
\end{array}$$

$$\Downarrow$$

Storage status of arrays A(LNA, M), JA(LNA, M)

$$\begin{array}{c}
\\
\text{LNA} \updownarrow
\end{array}
\begin{array}{c}
\text{A} \\
\begin{array}{|ccccc|}
\hline
a_{1,1} & a_{1,2} & a_{1,6} & a_{1,4} & * \\
a_{2,2} & a_{2,1} & a_{2,3} & a_{2,4} & a_{2,7} \\
a_{3,3} & a_{3,2} & a_{3,6} & * & * \\
\vdots & \vdots & \vdots & \vdots & \vdots \\
a_{6,6} & a_{6,1} & a_{6,3} & a_{6,4} & a_{6,7} \\
a_{7,7} & a_{7,2} & * & * & * \\
\hline
\multicolumn{5}{|c|}{<----\quad \text{M} \quad ---->} \\
\hline
\end{array}
\end{array}
\quad
\begin{array}{c}
\text{JA} \\
\begin{array}{|ccccc|}
\hline
1 & 2 & 6 & 4 & 0 \\
2 & 1 & 3 & 4 & 7 \\
3 & 2 & 6 & 0 & * \\
\vdots & \vdots & \vdots & \vdots & \vdots \\
6 & 1 & 3 & 4 & 7 \\
7 & 2 & 0 & * & * \\
\hline
\multicolumn{5}{|c|}{<----\quad \text{M} \quad ---->} \\
\hline
\end{array}
\end{array}
\begin{array}{c}
\\
\updownarrow \text{N}
\end{array}$$

Remarks

a. N is order of Matrix A.

b. LNA $\geq$ N must hold.

c. M is the column number of Array A, which contains the nonzero elements of Matrix A.

d. Array A should contain nonzero elements of Matrix A so that:

- Diagonal elements are stored in the first column.
- Nonzero elements in the lower triangular part and the upper triangular part are stored in the second though M-th columns, with the first one in the second column, one adjacent to the next in each row. Here, it is unnecessary that nonzero elements in each row are stored sequentially.
- Arbitrary values can be stored in the remaining positions that are marked with '$*$'.

e. Array JA should contain the column indices in Matrix A of those elements that correspond to the elements contained in Array A.

For those rows in which M $-$ 1 becomes greater than the number of nonzero elements in the lower and upper triangular part, value 0 should be stored in the right neighbor of the rightmost position of the region in JA in which the column indices of nonzero elements in Matrix A are stored. Arbitrary values can be stored in the remaining positions that are marked with '$*$'.

Figure B$-$16 Storage format for Asymmetric Random Sparse Matrix (ELLPACK Format)

(2) One-dimensional column-oriented list format

Matrix A to be stored

$$\begin{array}{|ccccccc|}
\hline
a_{1,1} & a_{1,2} & 0.0 & a_{1,4} & 0.0 & a_{1,6} & 0.0 \\
a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} & 0.0 & 0.0 & a_{2,7} \\
0.0 & a_{3,2} & a_{3,3} & 0.0 & 0.0 & a_{3,6} & 0.0 \\
a_{4,1} & a_{4,2} & 0.0 & a_{4,4} & a_{4,5} & a_{4,6} & 0.0 \\
0.0 & 0.0 & 0.0 & 0.0 & a_{5,5} & 0.0 & 0.0 \\
a_{6,1} & 0.0 & a_{6,3} & a_{6,4} & 0.0 & a_{6,6} & a_{6,7} \\
0.0 & a_{7,2} & 0.0 & 0.0 & 0.0 & 0.0 & a_{7,7} \\
\hline
\end{array}$$

$\Downarrow$

Storage status of arrays VALUES(NNZ), IRWIND(NNZ) and IPONTR(N+1)

VALUES	IRWIND	IPONTR
$a_{1,1}$	1	1
$a_{2,1}$	2	5
$a_{4,1}$	4	10
$a_{6,1}$	6	13
$a_{1,2}$	1	17
$a_{2,2}$	2	19
$a_{3,2}$	3	23
$a_{4,2}$	4	26
$a_{7,2}$	7	
$a_{2,3}$	2	
$a_{3,3}$	3	
$a_{6,3}$	6	
$a_{1,4}$	1	
$a_{2,4}$	2	
$a_{4,4}$	4	
$a_{6,4}$	6	
$a_{4,5}$	4	
$a_{5,5}$	5	
$a_{1,6}$	1	
$a_{3,6}$	3	
$a_{4,6}$	4	
$a_{6,6}$	6	
$a_{2,7}$	2	
$a_{6,7}$	6	
$a_{7,7}$	7	

NNZ

N+1

ITYPE = $\boxed{1}$

Remarks

a. N is the order of Matrix A.

b. NNZ is the number of nonzero elements.

c. VALUES contains the nonzero elements of Matrix A, stored sequentially beginning with the first column.

d. IRWIND contains the row indices in the original matrix A of the elements stored in VALUES.

e. IPONTR contains values equal to the positions in array VALUES of the diagonal elements. Here, IPONTR(1) contains the value 1 and IPONTR(N+1)contains the value NNZ+1.

Figure B−17 Storage of Asymmetric Random Sparse Matrix (One-dimensional Column-oriented List Format)

(3) JAD format (Jagged diagonals storage format)

Matrix A to be stored

$$\begin{array}{|ccccccc|}\hline a_{1,1} & a_{1,2} & 0. & a_{1,4} & 0. & a_{1,6} & 0. \\ a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} & 0. & 0. & a_{2,7} \\ 0. & a_{3,2} & a_{3,3} & 0. & 0. & a_{3,6} & 0. \\ a_{4,1} & a_{4,2} & 0. & a_{4,4} & a_{4,5} & a_{4,6} & 0. \\ 0. & 0. & 0. & a_{5,4} & a_{5,5} & 0. & 0. \\ a_{6,1} & 0. & a_{6,3} & a_{6,4} & 0. & a_{6,6} & 0. \\ 0. & a_{7,2} & 0. & 0. & 0. & 0. & a_{7,7} \\ \hline \end{array}$$

$\Downarrow$

Storage status within array JADORD

3	↑
1	
5	
2	N
6	
4	
7	↓

N

$\boxed{7}$

MJAD

$\boxed{5}$

Storage status within array IAJAD

← − − (MJAD + 1) − − →

1	8	15	20	24	26

Storage status within array AJAD

← − − − − − MJAD − − − − − →

$a_{2,1}$	$a_{2,2}$	$a_{2,3}$	$a_{2,4}$	$a_{2,7}$
$a_{4,1}$	$a_{4,2}$	$a_{4,4}$	$a_{4,5}$	$a_{4,6}$
$a_{1,1}$	$a_{1,2}$	$a_{1,4}$	$a_{1,6}$	
$a_{6,1}$	$a_{6,3}$	$a_{6,4}$	$a_{6,6}$	
$a_{3,2}$	$a_{3,3}$	$a_{3,6}$		
$a_{5,4}$	$a_{5,5}$			
$a_{7,2}$	$a_{7,7}$			

Storage status within array JAJAD

← − − − MJAD− →

1	2	3	4	7
1	2	4	5	6
1	2	4	6	
1	3	4	6	
2	3	6		
4	5			
2	7			

Remarks

a. The data types of matrix elements may be either real or complex.

b. N is the order of matrix A.

c. To obtain JAD storage of matrix A, consider a data arrangement as follows:
Push rowwise the whole nonzero elements of matrix A to the left side, then sort the rows with respect to the number of nonzero elements in descending order;
The columns in this arrangement are called **jagged diagonal**s. The number of jagged diagonals is stored in the parameter MJAD. The elements are stored in array AJAD "jagged diagonal"wise, successively from the leftmost jagged diagonal to the rightmost one.

d. The row indices of the elements stored in array AJAD are stored in array JAJAD.

e. The indices of the starting element of each jagged diagonal in array AJAD are stored in IAJAD. The number of elements stored in AJAD added by 1, is stored in the (MJAD+1)-th element of IAJAD.

f. The value 1 is set to IAJAD(1).

g. (The number of elements stored in arrays AJAD, JAJAD) = IAJAD(MJAD+1)−1.

h. In the figure above illustrates a JAD storage for a structurally symmetric matrix A. But naturally, this storage is even available for a general asymmetric matrix A.

Figure B−18 JAD format

(4) **One-dimensional row-oriented block list format**

Matrix A to be stored

$$\begin{array}{|cccc|}\hline B & 0 & C & 0 \\ D & F & 0 & G \\ 0 & 0 & H & 0 \\ 0 & K & 0 & L \\ \hline\end{array}$$

$$B = \begin{bmatrix} b_1 & b_2 \\ b_3 & b_4 \end{bmatrix},\quad C = \begin{bmatrix} c_1 & c_2 \\ c_3 & c_4 \end{bmatrix},\quad D = \begin{bmatrix} d_1 & d_2 \\ d_3 & d_4 \end{bmatrix},\quad F = \begin{bmatrix} f_1 & f_2 \\ f_3 & f_4 \end{bmatrix},\quad G = \overset{\leftarrow -\mathrm{M}- \rightarrow}{\begin{bmatrix} g_1 & g_2 \\ g_3 & g_4 \end{bmatrix}},\ \Big\updownarrow \mathrm{M} = 2$$

$$H = \begin{bmatrix} h_1 & h_2 \\ h_3 & h_4 \end{bmatrix},\quad K = \begin{bmatrix} k_1 & k_2 \\ k_3 & k_4 \end{bmatrix},\quad L = \begin{bmatrix} l_1 & l_2 \\ l_3 & l_4 \end{bmatrix},\quad 0 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

$$\Downarrow$$

Storage status within array A(LXA×M×M), JA(LXA), IA(NB+1)

	A	JA	IA	
↑	b_1	1	1	↑
	b_2	3	3	
	b_3	1	6	NB+1
	b_4	2	7	
	c_1	4	9	↓
	c_2	3		
	c_3	2		
	c_4	4		
LXA×M×M	⋮			
	k_1			
	k_2			
	k_3			
	k_4			
	l_1			
	l_2			
	l_3			
↓	l_4			

Remarks

a. The data types of matrix elements may be either real or complex.

b. NB is the number of block rows (or columns) for dividing matrix A into M×M block matrix.

c. LXA is the number of nonzero M×M block matrices.

d. A contains the nonzero block matrices of matrix A, stored sequentially beginning with the first block row.

e. JA contains the column block numbers in the original matrix A of the block matrices stored in A.

f. IA contains values equal to the positions in array A of the diagonal block matrices. The NB+1-th element contains the value LXA+1.

Figure B−19 One-dimensional Row-oriented Block List Format (for M = 2)

(5) **MJAD format (Multiple jagged diagonals storage format)**

Matrix A to be stored

$$\begin{array}{|cccc|}\hline B & 0 & C & 0 \\ D & F & 0 & G \\ 0 & 0 & H & 0 \\ 0 & K & 0 & L \\ \hline\end{array}$$

$$B = \begin{bmatrix} b_1 & b_2 \\ b_3 & b_2 \end{bmatrix},\ C = \begin{bmatrix} c_1 & c_2 \\ c_3 & c_4 \end{bmatrix},\ D = \begin{bmatrix} d_1 & d_2 \\ d_3 & d_4 \end{bmatrix},\ F = \begin{bmatrix} f_1 & f_2 \\ f_3 & f_4 \end{bmatrix},\ G = \overset{\leftarrow -\mathrm{M}- \rightarrow}{\begin{bmatrix} g_1 & g_2 \\ g_3 & g_4 \end{bmatrix}},\ \updownarrow \mathrm{M} = 2$$

$$H = \begin{bmatrix} h_1 & h_2 \\ h_3 & h_4 \end{bmatrix},\ K = \begin{bmatrix} k_1 & k_2 \\ k_3 & k_4 \end{bmatrix},\ L = \begin{bmatrix} l_1 & l_2 \\ l_3 & l_4 \end{bmatrix},\ 0 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

$$\Downarrow$$

NB $\boxed{4}$ MJAD $\boxed{3}$

Storage status within arrayJADORD

$$\begin{array}{|c|c}\hline 3 & \\ \hline 1 & \\ \hline 4 & \mathrm{NB} \\ \hline 2 & \\ \hline\end{array}$$

Storage status within arrayIAJAD

MJAD+1

$\leftarrow - - - - - \rightarrow$

1	5	8	9

($\star$)Storage status within arrayAJAD

$\leftarrow -\mathrm{M} \times \mathrm{M} - - \rightarrow$

$$\begin{array}{c|c|c|c|c|}\hline & d_1 & d_2 & d_3 & d_4 \\ & k_1 & k_2 & k_3 & k_4 \\ & b_1 & b_2 & b_3 & b_4 \\ \mathrm{LXA} & h_1 & h_2 & h_3 & h_4 \\ & f_1 & f_2 & f_3 & f_4 \\ & l_1 & l_2 & l_3 & l_4 \\ & c_1 & c_2 & c_3 & c_4 \\ & g_1 & g_2 & g_3 & g_4 \\ \hline\end{array}$$

Storage status within arrayJAJAD

$\leftarrow$ MJAD $\rightarrow$

1	2	4
2	4	
1	3	
3		

Remarks

a. The data types of matrix elements may be either real or complex.

b. NB is number of block rows (or columns) for dividing matrix A into M×M block matrix.

c. Push rowwise the whole nonzero block matrices of matrix A to the left side, then sort the rows with respect to the number of nonzero block matrix in descending order;
The block columns in this arrangement are called **jagged diagonals**. The number of jagged diagonals is stored in the parameter MJAD. The storage method for array AJAD is described as follows. The first row, first column elements among from each block matrix (D, K, B, H, F, C, G) are taken. Here, these elements are arranged in the same order as block matrices appear along the jagged diagonal above $(d_1, k_1, b_1, h_1, f_1, c_1, g_1)$. This method perform repeatedly until M-th row, M-th column elements are taken and stored in array AJAD.

d. The block column indices of the block array stored in array AJAD are stored in array JAJAD

e. The indices of the starting element of each jagged diagonal in array AJAD stored in array IAJAD. The number of block array stored in AJAD added by 1, is stored in the MJAD+1-th element of AJAD.

f. Each elements in the same block will be arranged with equal intervals of LXA in memory ($\star$). For example, elements in the block D (d_1, d_2, d_3, d_4) will be arranged with equal intervals of LXA in memory.

g. The value 1 is set to IAJAD(1).

h. (The number of elements stored in MJAD) = (IAJAD(MJAD+1)−1)×M×M

Figure B−20 MJAD format(for M = 2)

B.2.11 Hermitian sparse matrix (Hermitian one-dimensional row-oriented list type) (upper triangular type)

Matrix A to be stored

$$\begin{array}{|ccccccc|}
\hline
a_{1,1} & a_{1,2} & 0. & a_{1,4} & 0. & a_{1,6} & 0. \\
\overline{a_{1,2}} & a_{2,2} & a_{2,3} & a_{2,4} & 0. & 0. & a_{2,7} \\
0. & \overline{a_{2,3}} & a_{3,3} & 0. & 0. & a_{3,6} & 0. \\
\overline{a_{1,4}} & \overline{a_{2,4}} & 0. & a_{4,4} & a_{4,5} & a_{4,6} & 0. \\
0. & 0. & 0. & \overline{a_{4,5}} & a_{5,5} & 0. & 0. \\
\overline{a_{1,6}} & 0. & \overline{a_{3,6}} & \overline{a_{4,6}} & 0. & a_{6,6} & 0. \\
0. & \overline{a_{2,7}} & 0. & 0. & 0. & 0. & a_{7,7} \\
\hline
\end{array}$$

$$\Downarrow$$

N	NA
7	16

Storage status within arrays A(NA), JA(NA), IA(N+1)

	A	JA	IA	
NA	$a_{1,1}$	1	1	N+1
	$a_{1,2}$	2	5	
	$a_{1,4}$	4	9	
	$a_{1,6}$	6	11	
	$a_{2,2}$	2	14	
	$a_{2,3}$	3	15	
	$a_{2,4}$	4	16	
	$a_{2,7}$	7	17	
	$a_{3,3}$	3		
	$a_{3,6}$	6		
	$a_{4,4}$	4		
	$a_{4,5}$	5		
	$a_{4,6}$	6		
	$a_{5,5}$	5		
	$a_{6,6}$	6		
	$a_{7,7}$	7		

Remarks

a. The $\overline{x}$ indicates the complex conjugate of x.

b. N is order of the matrix A.

c. NA is the number of the diagonal elements and the nonzero elements in the upper triangle of the matrix A.

d. Array A contains the diagonal elements and the nonzero elements in the upper triangle of the matrix A, stored sequentially beginning with the first row.

e. Array JA contains the column numbers in the original matrix A of the elements stored in A.

f. Array IA contains values equal to the positions in array A of the diagonal element in each row. The N+1-th element contains the value NA+1.

g. $1 \leq \mathrm{N} \leq \mathrm{NA}$ must be satisfied.

Figure B−21 Storage format for Hermitian Sparse Matrix (Hermitian One-dimensional Row-oriented List Type)

Appendix C

MACHINE CONSTANTS USED IN ASL

C.1 Units for Determining Error

The table below shows values in ASL 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 uses these units for determining convergence and zeros.

Table C−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.

C.2 Maximum and Minimum Values of Floating Point Data

The table below shows maximum and minimum values of floating point data defined within ASL. 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 C−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

(*) DMP Functions: Distributed Memory Parallel Functions

(*) SMP Functions: Shared Memory Parallel Functions