NAG Library Chapter Introduction

F06 – Linear Algebra Support Routines

Contents

1 Scope of the Chapter

This chapter is concerned with basic linear algebra routines which perform elementary algebraic operations involving scalars, vectors and matrices. It includes routines which conform to the specifications of the BLAS (Basic Linear Algebra Subprograms).

2 Background to the Problems

A number of the routines in this chapter meet the specification of the Basic Linear Algebra Subprograms (BLAS) as described in Lawson *et al.* (1979), Dodson *et al.* (1991), Dongarra *et al.* (1988) and Dongarra *et al.* (1990). The first reference describes a set of routines concerned with operations on scalars and vectors: these will be referred to here as the Level-0 and the Level-1 BLAS; the second reference describes a set of routines concerned with operations on sparse vectors: these will be referred to here as the Level-1 Sparse BLAS; the third reference describes a set of routines concerned with matrix-vector operations: these will be referred to here as the Level-2 BLAS; and the fourth reference describes a set of routines concerned with matrix-matrix operations: these will be referred to here as the Level-3 BLAS.

More generally we refer to the scalar routines in the chapter as Level-0 routines, to the vector routines as Level-1 routines, to the matrix-vector and matrix routines as Level-2 routines, and to the matrix-matrix routines as Level-3 routines. The terminology reflects the number of operations involved. For example, a Level-2 routine involves $O(n^2)$ operations for an $n \times n$ matrix.

2.1 The Use of BLAS Names

Many of the routines in other chapters of the Library call the routines in this chapter, and in particular a number of the BLAS are called. These routines are usually called by the BLAS name and so, for correct operation of the Library, it is essential that you do not attempt to link your own versions of these routines. If you are in any doubt about how to avoid this, please consult your computer centre or the NAG Response Centre.

The BLAS names are used in order to make use of efficient implementations of the routines when these exist. Such implementations are stringently tested before being used, to ensure that they correctly meet the specification of the BLAS, and that they return the desired accuracy (see, for example, Dodson *et al.* (1991), Dongarra *et al.* (1988) and Dongarra *et al.* (1990)).

2.2 Background Information

Most of the routines in this chapter implement straightforward scalar, vector and matrix operations that need no further explanation beyond a statement of the purpose of the routine. In this section we give some additional background information to those few cases where additional explanation may be necessary. A sub-section is devoted to each topic.

2.2.1 Real plane rotations

There are a number of routines in the chapter concerned with setting up and applying plane rotations. This section discusses the real case and the next section looks at the complex case. For further background information see Golub and Van Loan (1996).

A plane rotation matrix for the (i,j) plane, R_{ij}, is an orthogonal matrix that is different from the unit matrix only in the elements r_{ii}, r_{jj}, r_{ij} and r_{ji}. If we put

$$R = \begin{pmatrix} r_{ii} & r_{ij} \\ r_{ji} & r_{jj} \end{pmatrix}, \qquad (1)$$

then, in the real case, it is usual to choose R_{ij} so that

$$R = \begin{pmatrix} c & s \\ -s & c \end{pmatrix}, \quad c = \cos\theta, \quad s = \sin\theta.$$

An exception is routine F06FPF which applies the so-called symmetric rotation for which

$$R = \begin{pmatrix} c & s \\ s & -c \end{pmatrix}. \tag{2}$$

The application of plane rotations is straightforward and needs no further elaboration, so further comment is made only on the construction of plane rotations.

The most common use of plane rotations is to choose c and s so that for given a and b,

$$\begin{pmatrix} c & s \\ -s & c \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} d \\ 0 \end{pmatrix}. \tag{3}$$

In such an application the matrix R is often termed a **Givens rotation** matrix. There are two approaches to the construction of real Givens rotations in Chapter F06.

The BLAS routine F06AAF (DROTG), see Lawson *et al.* (1979) and Dodson and Grimes (1982), computes c, s and d as

$$d = \sigma\left(a^2 + b^2\right)^{1/2},$$
$$c = \begin{cases} a/d, & d \neq 0, \\ 1, & d = 0, \end{cases} \qquad s = \begin{cases} b/d, & d \neq 0, \\ 0, & d = 0, \end{cases} \tag{4}$$

where $\sigma = \begin{cases} \operatorname{sign} a, & |a| > |b| \\ \operatorname{sign} b, & |a| \leq |b| \end{cases}$.

The value z defined as

$$z = \begin{cases} s, & |s| < c \quad \text{or} \quad c = 0 \\ 1/c, & 0 < |c| \leq s \end{cases} \tag{5}$$

is also computed and this enables c and s to be reconstructed from the single value z as

$$c = \begin{cases} 0, & z = 1 \\ \left(1 - z^2\right)^{1/2}, & |z| < 1 \\ 1/z, & |z| > 1 \end{cases} \qquad s = \begin{cases} 1, & z = 1 \\ z, & |z| < 1 \\ \left(1 - c^2\right)^{1/2}, & |z| > 1 \end{cases}.$$

The other Chapter F06 routines for constructing Givens rotations are based on the computation of the tangent, $t = \tan\theta$. t is computed as

$$t = \begin{cases} 0, & b = 0 \\ b/a, & |b| \leq |a|.\mathit{flmax}, b \neq 0 \\ \operatorname{sign}(b/a).\mathit{flmax}, & |b| > |a|.\mathit{flmax} \\ \operatorname{sign}(b).\mathit{flmax}, & b \neq 0, a = 0 \end{cases} \tag{6}$$

where $\mathit{flmax} = 1/\mathit{flmin}$ and flmin is the small positive value returned by X02AMF. The values of c and s are then computed or reconstructed via t as

$$c = \begin{cases} 1/\left(1 + t^2\right)^{1/2}, & \sqrt{\text{eps}} \leq |t| \leq 1/\sqrt{\text{eps}} \\ 1, & |t| < \sqrt{\text{eps}} \\ 1/|t|, & |t| > 1/\sqrt{\text{eps}} \end{cases} \qquad s = \begin{cases} c.t, & \sqrt{\text{eps}} \leq |t| \leq 1/\sqrt{\text{eps}} \\ t, & |t| < \sqrt{\text{eps}} \\ \operatorname{sign} t, & |t| > 1/\sqrt{\text{eps}} \end{cases} \tag{7}$$

where eps is the ***machine precision***. Note that c is always non-negative in this scheme and that the same expressions are used in the initial computation of c and s from a and b as in any subsequent recovery of c and s via t. This is the approach used by many of the NAG Library routines that require plane rotations. d is computed simply as

$$d = c.a + s.b.$$

You need not be too concerned with the above detail, since routines are provided for setting up, recovering and applying such rotations.

Another use of plane rotations is to choose c and s so that for given x, y and z

$$\begin{pmatrix} c & s \\ -s & c \end{pmatrix} \begin{pmatrix} x & y \\ y & z \end{pmatrix} \begin{pmatrix} c & -s \\ s & c \end{pmatrix} = \begin{pmatrix} a & 0 \\ 0 & b \end{pmatrix}. \quad (8)$$

In such an application the matrix R is often termed a **Jacobi rotation** matrix. The routine that generates a Jacobi rotation (F06BEF) first computes the tangent t and then computes c and s via t as described above for the Givens rotation.

2.2.2 Complex plane rotations

In the complex case a plane rotation matrix for the (i,j) plane, R_{ij} is a unitary matrix and, analogously to the real case, it is usual to choose R_{ij} so that

$$R = \begin{pmatrix} \bar{c} & \bar{s} \\ -s & c \end{pmatrix}, \quad |c|^2 + |s|^2 = 1, \quad (9)$$

where $\bar{a}$ denotes the complex conjugate of a.

The BLAS (see Lawson *et al.* (1979)) do not contain a routine for the generation of complex rotations, and so the routines in Chapter F06 are all based upon computing c and s via $t = b/a$ in an analogous manner to the real case. R can be chosen to have either c real, or s real and there are routines for both cases.

When c is real then it is non-negative and the transformation

$$\begin{pmatrix} c & \bar{s} \\ -s & c \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} d \\ 0 \end{pmatrix} \quad (10)$$

is such that if a is real then d is also real.

When s is real then the transformation

$$\begin{pmatrix} \bar{c} & s \\ -s & c \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} d \\ 0 \end{pmatrix} \quad (11)$$

is such that if b is real then d is also real.

2.2.3 Elementary real (Householder) reflections

There are a number of routines in the chapter concerned with setting up and applying Householder transformations. This section discusses the real case and the next section looks at the complex case. For further background information see Golub and Van Loan (1996).

A real elementary reflector, P, is a matrix of the form

$$P = I - \mu u u^{\mathrm{T}}, \quad \mu u^{\mathrm{T}} u = 2, \quad (12)$$

where μ is a scalar and u is a vector, and P is both symmetric and orthogonal. In the routines in Chapter F06, u is expressed in the form

$$u = \begin{pmatrix} \zeta \\ z \end{pmatrix}, \quad \zeta \text{ a scalar} \quad (13)$$

because in many applications ζ and z are not contiguous elements. The usual use of elementary reflectors is to choose μ and u so that for given α and x

$$P\begin{pmatrix} \alpha \\ x \end{pmatrix} = \begin{pmatrix} \beta \\ 0 \end{pmatrix}, \quad \alpha \text{ and } \beta \text{ scalars.} \quad (14)$$

Such a transformation is often termed a **Householder transformation**. There are two choices of μ and u available in Chapter F06.

The first form of the Householder transformation is compatible with that used by LINPACK (see Dongarra *et al.* (1979)) and has

$$\mu = 1/\zeta. \tag{15}$$

This choice makes ζ satisfy

$$1 \leq \zeta \leq 2.$$

The second form, and the form used by many of the NAG Library routines, has

$$\mu = 1 \tag{16}$$

which makes

$$1 \leq \zeta \leq \sqrt{2}.$$

In both cases the special setting

$$\zeta = 0 \tag{17}$$

is used by the routines to flag the case where $P = I$.

Note that while there are routines to apply an elementary reflector to a vector, there are no routines available in Chapter F06 to apply an elementary reflector to a matrix. This is because such transformations can readily and efficiently be achieved by calls to the matrix-vector Level 2 BLAS routines. For example, to form PA for a given matrix

$$\begin{aligned} PA &= (I - \mu uu^{\mathrm{T}})A = A - \mu uu^{\mathrm{T}}A \\ &= A - \mu ub^{\mathrm{T}}, \quad b = A^{\mathrm{T}}u, \end{aligned} \tag{18}$$

and so we can call a matrix-vector product routine to form $b = A^{\mathrm{T}}u$ and then call a rank-one update routine to form $(A - \mu ub^{\mathrm{T}})$. Of course, we must skip the transformation when ζ has been set to zero.

2.2.4 Elementary complex (Householder) reflections

A complex elementary reflector, P, is a matrix of the form

$$P = I - \mu uu^{\mathrm{H}}, \quad \mu u^{\mathrm{H}}u = 2, \quad \mu \text{ real},$$

where u^{H} denotes the complex conjugate of u^{T}, and P is both Hermitian and unitary. For convenience in a number of applications this definition can be generalized slightly by allowing μ to be complex and so defining the generalized elementary reflector as

$$P = I - \mu uu^{\mathrm{H}}, \quad |\mu|^2 u^{\mathrm{H}}u = \mu + \bar{\mu} \tag{19}$$

for which P is still unitary, but is no longer Hermitian.

The Chapter F06 routines choose μ and ζ so that

$$\mathrm{Re}(\mu) = 1, \quad \mathrm{Im}(\zeta) = 0 \tag{20}$$

and this reduces to (12) with the choice (16) when μ and u are real. This choice is used because μ and u can now be chosen so that in the Householder transformation (14) we can make

$$\mathrm{Im}(\beta) = 0$$

and, as in the real case,

$$1 \leq \zeta \leq \sqrt{2}.$$

Rather than returning μ and ζ as separate arguments the Chapter F06 routines return the single complex value θ defined as

$$\theta = \zeta + i.\,\mathrm{Im}(\mu), \quad i = \sqrt{-1}.$$

Obviously ζ and μ can be recovered as

$$\zeta = \mathrm{Re}(\theta), \quad \mu = 1 + i.\,\mathrm{Im}(\theta).$$

The special setting

$$\theta = 0$$

is used to flag the case where $P = I$, and

$$\mathrm{Re}(\theta) \leq 0, \quad \mathrm{Im}(\theta) \neq 0$$

is used to flag the case where

$$P = \begin{pmatrix} \gamma & 0 \\ 0 & I \end{pmatrix}, \quad \gamma \text{ a scalar} \tag{21}$$

and in this case θ actually contains the value of γ. Notice that with both (18) and (21) we merely have to supply $\bar{\theta}$ rather than θ in order to represent P^{H}.

3 Recommendations on Choice and Use of Available Routines

3.1 Naming Scheme

3.1.1 NAG names

Table 1 shows the naming scheme for the routines in this chapter.

		Level-0	Level-1	Level-2	Level-3
integer	Chapter F06 routine	–	F06D_F	–	–
‘real’	BLAS routine	F06A_F	F06E_F	F06P_F	F06Y_F
‘real’	Chapter F06 routine	F06B_F	F06F_F	F06Q_F F06R_F	–
‘complex’	BLAS routine	–	F06G_F	F06S_F	F06Z_F
‘complex’	Chapter F06 routine	F06C_F	F06H_F	F06T_F F06U_F	–
‘mixed type’	BLAS routine	–	F06J_F	–	–
‘mixed type’	Chapter F06 routine	–	F06K_F	F06V_F	–
‘real’ and ‘complex’	LAPACK routines	–	–	F06W_F	F06W_F

Table 1

The heading ‘mixed type’ is for routines where a mixture of data types is involved, such as a routine that returns the real Euclidean length of a complex vector. In future marks of the Library, routines may be included in categories that are currently empty and further categories may be introduced.

3.1.2 BLAS names

Those routines which conform to the specifications of the BLAS may be called either by their NAG names or by their BLAS names.

In many implementations of the NAG Library, references to BLAS names may be linked to an efficient machine-specific implementation of the BLAS, usually provided by the vendor of the machine. Such implementations are stringently tested before being used with the NAG Library, to ensure that they correctly meet the specifications of the BLAS, and that they return the desired accuracy. Use of BLAS names is recommended for efficiency.

References to NAG routine names (beginning F06-) are always linked to the code provided in the NAG Library and may be significantly slower than the equivalent BLAS routine.

The names of the Level-2 and Level-3 BLAS follow a simple scheme (which is similar to that used for LAPACK routines in Chapters F07 and F08). Each name has the structure **XYYZZZ**, where the components have the following meanings:

– the initial letter **X** indicates the data type (real or complex) and precision:

S real, single precision (in Fortran, `REAL`)

D real, double precision (in Fortran, `DOUBLE PRECISION`)

C complex, single precision (in Fortran, `COMPLEX`)

Z complex, double precision (in Fortran, `COMPLEX*16` or `DOUBLE COMPLEX`)

– the second and third letters **YY** indicate the type of the matrix A (and in some cases its storage scheme):

GE general

GB general band

SY symmetric

SP symmetric (packed storage)

SB symmetric band

HE (complex) Hermitian

HP (complex) Hermitian (packed storage)

HB (complex) Hermitian band

TR triangular

TP triangular (packed storage)

TB triangular band

– the remaining 1, 2 or 3 letters **ZZZ** indicate the computation performed:

MV matrix-vector product

MM matrix-matrix product

R rank-1 update

R2 rank-2 update

RK rank-k update

R2K rank-$2k$ update

SV solve a system of linear equations

SM solve a system of linear equations with a matrix of right-hand sides

Thus the routine DGEMV performs a matrix-vector product involving a real general matrix in double precision; the corresponding routine for a complex general matrix is ZGEMV.

The names of the Level-1 BLAS mostly follow the same convention for the initial letter (S-, C-, D- or Z-), except for a few involving data of mixed type, where the first two characters are precision-dependent.

3.1.3 LAPACK names

There are some LAPACK routines in this chapter that have BLAS-like functionalty. Four are equivalent to BLAS routines but for matrices stored in Rectangular Full Packed (RFP) format. The naming convention for these is as above with the addition of the matrix types:

HF (complex) Hermitian (RFP storage)

TF triangular (RFP storage)

SF symmetric (RFP storage)

There are an additonal two that compute norms of RFP matrices. These have second and third letters LA (signifying LAPACK), fourth letter N (signifying norm), and fifth and sixth letter signifying matrix type as above. For example ZLANHF computes the norm of a Hermitian matrix in RFP format.

3.2 The Level-0 Scalar Routines

The Level-0 routines perform operations on scalars or on vectors or matrices of order 2.

3.3 The Level-1 Vector Routines

The Level-1 routines perform operations either on a single vector or on a pair of vectors.

3.4 The Level-2 Matrix-vector and Matrix Routines

The Level-2 routines perform operations involving either a matrix on its own, or a matrix and one or more vectors.

3.5 The Level-3 Matrix-matrix Routines

The Level-3 routines perform operations involving matrix-matrix products.

3.6 Vector Arguments

Vector arguments (except in the Level-1 Sparse BLAS) are represented by a one-dimensional array, immediately followed by an **increment** argument whose name consists of the three characters INC followed by the name of the array. For example, a vector x is represented by the two arguments X and INCX. The length of the vector, n say, is passed as a separate argument, N.

The increment argument is the spacing (stride) in the array between the elements of the vector. For instance, if $\mathrm{INCX} = 2$, then the elements of x are in locations $x(1), x(3), \ldots, x(2n-1)$ of the array X and the intermediate locations $x(2), x(4), \ldots, x(2n-2)$ are not referenced.

When $\mathrm{INCX} > 0$, the vector element x_i is in the array element $\mathrm{X}(1 + (i-1) \times \mathrm{INCX})$. When $\mathrm{INCX} \leq 0$, the elements are stored in the reverse order so that the vector element x_i is in the array element $\mathrm{X}(1 - (n-i) \times \mathrm{INCX})$ and hence, in particular, the element x_n is in $\mathrm{X}(1)$. The declared length of the array X in the calling subroutine must be at least $(1 + (\mathrm{N} - 1) \times |\mathrm{INCX}|)$.

Negative increments are permitted only for:

Level-1 routines which have more than one vector argument;

Level-2 BLAS routines (but not for other Level-2 routines)

Zero increments are formally permitted for Level-1 routines with more than one argument (in which case the element $\mathrm{X}(1)$ is accessed repeatedly), but their use is strongly discouraged since the effect may be implementation-dependent. There is usually an alternative routine in this chapter, with a simplified argument list, to achieve the required purpose. Zero increments are not permitted in the Level-2 BLAS.

In the Level-1 Sparse BLAS, each routine operates on two vectors x and y. The vector x is stored as a compressed sparse vector, and is represented by the three arguments NZ, X and INDX; NZ is the number of 'interesting' (usually nonzero) elements of x, and INDX is a one-dimensional **index** array such that

$$x(\mathrm{INDX}(k)) = \mathrm{X}(k), \quad k = 1, 2, \ldots, \mathrm{NZ}.$$

The (mathematical) length of the vector, n say, does not need to be supplied; it is assumed that $1 \leq \mathrm{INDX}(k) \leq n$. For example, the vector

$$x = (0, 4, 0, 0, 1, 0, 0, 0, 6, 0)$$

could be represented with $\mathrm{NZ} = 3$, $\mathrm{X} = (4, 1, 6)$, $\mathrm{INDX} = (2, 5, 9)$. The second vector y is stored conventionally, and is represented simply by the one-dimensional array Y, with y_i in $Y(i)$; the increment is assumed to be 1. Only the elements $Y(\mathrm{INDX}(k))$ are referenced.

Non-positive values of NZ are permitted, in which case the routines return immediately — except that functions set their value to zero before returning. For those routines where Y is an output argument **the values in the array INDX must be distinct**; violating this condition may yield incorrect results.

3.7 Matrix Arguments and Storage Schemes

In this chapter the following different storage schemes are used for matrices:

– **conventional storage** in a two-dimensional array;

– **packed and RFP storage** for symmetric, Hermitian or triangular matrices;

– **band storage** for band matrices;

– storage for spiked matrices.

These storage schemes are compatible with those used in Chapters F07 and F08. (Different schemes for packed or band storage are used in a few older routines in Chapters F01, F02, F03 and F04.)

Chapter F01 provides some utility routines for conversion between storage schemes.

In the examples, $*$ indicates an array element which need not be set and is not referenced by the routines. The examples illustrate only the relevant leading rows and columns of the arrays; array arguments may of course have additional rows or columns, according to the usual rules for passing array arguments in Fortran.

3.7.1 Conventional storage

Please see Section 3.3.1 in the F07 Chapter Introduction for full details.

3.7.2 Packed storage

Please see Section 3.3.2 in the F07 Chapter Introduction for full details.

3.7.3 Rectangular Full Packed (RFP) storage

Please see Section 3.3.3 in the F07 Chapter Introduction for full details.

3.7.4 Band storage

Please see Section 3.3.4 in the F07 Chapter Introduction for full details.

3.7.5 Unit triangular matrices

Please see Section 3.3.5 in the F07 Chapter Introduction for full details.

3.7.6 Real diagonal elements of complex Hermitian matrices

Please see Section 3.3.6 in the F07 Chapter Introduction for full details.

3.7.7 Spiked matrices

A few routines in this chapter (F06QSF, F06QWF, F06TSF and F06TWF) deal with **upper spiked** matrices. These are upper triangular matrices with an additional nonzero row or column below the diagonal.

The position of the spike is defined by indices k_1 and k_2; it is assumed that $k_1 < k_2$. A **row spike** has nonzero elements in the k_2th row, $a_{k_2,k}$ for $k = k_1, k_1 + 1, \ldots, k_2 - 1$; a **column spike** has nonzero elements in the k_1th column, a_{k+1,k_1} for $k = k_1, k_1 + 1, \ldots, k_2 - 1$. For example, when $n = 6$, $k_1 = 2$ and $k_2 = 5$:

Row spike	Column spike
$\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} & a_{16} \\ & a_{22} & a_{23} & a_{24} & a_{25} & a_{26} \\ & & a_{33} & a_{34} & a_{35} & a_{36} \\ & & & a_{44} & a_{45} & a_{46} \\ & a_{52} & a_{53} & a_{54} & a_{55} & a_{56} \\ & & & & & a_{66} \end{pmatrix}$	$\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} & a_{16} \\ & a_{22} & a_{23} & a_{24} & a_{25} & a_{26} \\ & a_{32} & a_{33} & a_{34} & a_{35} & a_{36} \\ & a_{42} & & a_{44} & a_{45} & a_{46} \\ & a_{52} & & & a_{55} & a_{56} \\ & & & & & a_{66} \end{pmatrix}$

The storage scheme adopted by the routines in this chapter is for the upper triangular part of the spiked matrix to be stored conventionally in a two-dimensional array A, with the subdiagonal elements of the spike stored in a separate vector.

3.8 Option Parameters

Many of the routines in this chapter have one or more **option arguments**, of type CHARACTER. The descriptions in the routine documents refer only to upper-case values (for example UPLO = 'U' or UPLO = 'L'); however, in every case, the corresponding lower-case characters may be supplied (with the same meaning). Any other value is illegal.

A longer character string can be passed as the actual argument, making the calling program more readable, but only the first character is significant. (This is a feature of Fortran.) For example:

```
CALL DTRSV('Upper','Transpose','Non-unit',...)
```

The following option arguments are used in this chapter:

If TRANS = 'N', operate with the matrix (Not transposed);

if TRANS = 'T', operate with the Transpose of the matrix;

if TRANS = 'C', operate with the Conjugate transpose of the matrix.

If UPLO = 'U', upper triangle or trapezoid of matrix;

if UPLO = 'L', lower triangle or trapezoid of matrix.

If DIAG = 'U', unit triangular;

if DIAG = 'N', nonunit triangular.

If SIDE = 'L', operate from the left-hand side;

if SIDE = 'R', operate from the right-hand side.

If PIVOT = 'V', variable pivot (in applying a sequence of plane rotations);

if PIVOT = 'B', bottom pivot;

if PIVOT = 'T', top pivot;

if PIVOT = 'F', fixed pivot.

If DIRECT = 'B', backward sequence of plane rotations;

if DIRECT = 'F', forward sequence of plane rotations.

If NORM = '1' or 'O', 1-norm of a matrix;

if NORM = 'I', ∞-norm of a matrix;

if NORM = 'F' or 'E', Frobenius or Euclidean norm of a matrix;

if NORM = 'M', maximum absolute value of the elements of a matrix (not strictly a norm).

If MATRIX = 'G', general (rectangular or square) matrix;

if MATRIX = 'U', upper trapezoidal or triangular matrix;

if MATRIX = 'L', lower trapezoidal or triangular matrix.

if TRANSR = 'N', matrix stored in normal RFP format (Not transposed).

if TRANSR = 'T', transpose of the matrix stored in RFP format.

if TRANSR = 'C', conjugate transpose of the matrix stored in RFP format.

3.8.1 Matrix norms

The option argument NORM specifies different matrix norms whose definitions are given here for reference (for a general m by n matrix A):

One-norm (NORM = 'O' or '1'):

$$\|A\|_1 = \max_j \sum_{i=1}^{m} |a_{ij}|;$$

Infinity-norm (NORM = 'I'):

$$\|A\|_\infty = \max_i \sum_{j=1}^{n} |a_{ij}|;$$

Frobenius or Euclidean norm (NORM = 'F' or 'E'):

$$\|A\|_F = \left(\sum_{i=1}^{m} \sum_{j=1}^{n} |a_{ij}|^2 \right)^{1/2}.$$

If A is symmetric or Hermitian, $\|A\|_1 = \|A\|_\infty$.

The argument NORM can also be used to specify the maximum absolute value $\max_{i,j}|a_{ij}|$ (if NORM = 'M'), but this is not a norm in the strict mathematical sense.

3.9 Error Handling

Routines in this chapter do not use the usual NAG Library error-handling mechanism, involving the argument IFAIL.

If one of the Level-2 or Level-3 BLAS routines is called with an invalid value of one of its arguments, then an error message is output on the error message unit (see X04AAF), giving the name of the routine and the number of the first invalid argument, and execution of the program is terminated. The following values of arguments are invalid:

– any value of the character arguments TRANS, TRANSA, TRANSB, UPLO, SIDE or DIAG, whose meaning is not specified;

– a negative value of any of the arguments M, N, K, KL or KU;

– too small a value for any of the leading dimension arguments;

– a zero value for the increment arguments INCX and INCY.

Zero values for the matrix dimensions M, N or K are considered valid.

The other routines in this chapter do not report any errors in their arguments. Normally, if called, for example, with an unspecified value for one of the option arguments, or with a negative value of one of the problem dimensions M or N, they simply do nothing and return immediately.

4 Functionality Index

Level 0 (Scalar) operations,
  complex numbers,
    apply similarity rotation to 2 by 2 Hermitian matrix F06CHF
    generate a plane rotation, storing the tangent, real cosine...................... F06CAF
    generate a plane rotation, storing the tangent, real sine F06CBF
    quotient of two numbers, with overflow flag.. F06CLF
    recover cosine and sine from given tangent, real cosine F06CCF
    recover cosine and sine from given tangent, real sine............................ F06CDF
  real numbers,
    apply similarity rotation to 2 by 2 symmetric matrix............................ F06BHF
    compute $(a2 + b2)1/2$... F06BNF
    compute Euclidean norm from scaled form F06BMF
    eigenvalue of 2 by 2 symmetric matrix... F06BPF
    generate a Jacobi plane rotation... F06BEF
    generate a plane rotation... F06AAF (DROTG)
    generate a plane rotation storing the tangent F06BAF

- rank-2 update,
  - Hermitian matrix — F06SRF (ZHER2)
  - Hermitian packed matrix — F06SSF (ZHPR2)
  - matrix copy, rectangular or trapezoidal — F06TFF
- solution of a system of equations,
  - triangular band matrix — F06SKF (ZTBSV)
  - triangular matrix — F06SJF (ZTRSV)
  - triangular packed matrix — F06SLF (ZTPSV)
- unitary similarity transformation of a Hermitian matrix,
  - as sequence of plane rotations — F06TMF

real matrix and vector(s),

- apply sequence of plane rotations to a rectangular matrix — F06QXF
- compute a norm or the element of largest absolute value,
  - band matrix — F06RBF
  - general matrix — F06RAF
  - Hessenberg matrix — F06RMF
  - matrix initialization — F06QHF
  - symmetric band matrix — F06REF
  - symmetric matrix — F06RCF
  - symmetric matrix, packed form — F06RDF
  - symmetric matrix, RFP format — F06WAF (DLANSF)
  - symmetric tridiagonal matrix — F06RPF
  - trapezoidal matrix — F06RJF
  - triangular band matrix — F06RLF
  - triangular matrix, packed form — F06RKF
  - tridiagonal matrix — F06RNF
- compute upper Hessenberg matrix by applying sequence of plane rotations to an upper triangular matrix — F06QVF
- compute upper spiked matrix by applying sequence of plane rotations to an upper triangular matrix — F06QWF
- matrix-vector product,
  - rectangular band matrix — F06PBF (DGBMV)
  - rectangular matrix — F06PAF (DGEMV)
  - symmetric band matrix — F06PDF (DSBMV)
  - symmetric matrix — F06PCF (DSYMV)
  - symmetric packed matrix — F06PEF (DSPMV)
  - triangular band matrix — F06PGF (DTBMV)
  - triangular matrix — F06PFF (DTRMV)
  - triangular packed matrix — F06PHF (DTPMV)
- orthogonal similarity transformation of a symmetric matrix,
  - as sequence of plane rotations — F06QMF
- permute rows or columns of a matrix,
  - permutations represented by an integer array — F06QJF
  - permutations represented by a real array — F06QKF
- QR factorization by sequence of plane rotations,
  - of rank-1 update of upper triangular matrix — F06QPF
  - of upper triangular matrix augmented by a full row — F06QQF
- QR factorization of UZ or RQ factorization of ZU, where U is upper triangular and Z is a sequence of plane rotations — F06QTF
- QR or RQ factorization by sequence of plane rotations,
  - of upper Hessenberg matrix — F06QRF
  - of upper spiked matrix — F06QSF
- rank-1 update,
  - rectangular matrix — F06PMF (DGER)
  - symmetric matrix — F06PPF (DSYR)
  - symmetric packed matrix — F06PQF (DSPR)
- rank-2 update,
  - matrix copy, rectangular or trapezoidal — F06QFF
  - symmetric matrix — F06PRF (DSYR2)

BLAS Routines

Real Matrices		
single precision	double precision	NAG
ISAMAX	IDAMAX	F06JLF (IDAMAX)
SASUM	DASUM	F06EKF (DASUM)
SAXPY	DAXPY	F06ECF (DAXPY)
SAXPYI	DAXPYI	F06ETF (DAXPYI)
SCASUM	DZASUM	F06JKF (DZASUM)
SCNRM2	DZNRM2	F06JJF (DZNRM2)
SCOPY	DCOPY	F06EFF (DCOPY)
SDOT	DDOT	F06EAF (DDOT)
SDOTI	DDOTI	F06ERF (DDOTI)
SGBMV	DGBMV	F06PBF (DGBMV)
SGEMM	DGEMM	F06YAF (DGEMM)
SGEMV	DGEMV	F06PAF (DGEMV)
SGER	DGER	F06PMF (DGER)
SGTHR	DGTHR	F06EUF (DGTHR)
SGTHRZ	DGTHRZ	F06EVF (DGTHRZ)
SNRM2	DNRM2	F06EJF (DNRM2)
SROT	DROT	F06EPF (DROT)
SROTG	DROTG	F06AAF (DROTG)
SROTI	DROTI	F06EXF (DROTI)
SSBMV	DSBMV	F06PDF (DSBMV)
SSCAL	DSCAL	F06EDF (DSCAL)
SSCTR	DSCTR	F06EWF (DSCTR)
SSPMV	DSPMV	F06PEF (DSPMV)
SSPR	DSPR	F06PQF (DSPR)
SSPR2	DSPR2	F06PSF (DSPR2)
SSWAP	DSWAP	F06EGF (DSWAP)
SSYMM	DSYMM	F06YCF (DSYMM)
SSYMV	DSYMV	F06PCF (DSYMV)
SSYR	DSYR	F06PPF (DSYR)
SSYR2	DSYR2	F06PRF (DSYR2)
SSYR2K	DSYR2K	F06YRF (DSYR2K)
SSYRK	DSYRK	F06YPF (DSYRK)
STBMV	DTBMV	F06PGF (DTBMV)
STBSV	DTBSV	F06PKF (DTBSV)
STPMV	DTPMV	F06PHF (DTPMV)
STPSV	DTPSV	F06PLF (DTPSV)
STRMM	DTRMM	F06YFF (DTRMM)
STRMV	DTRMV	F06PFF (DTRMV)
STRSM	DTRSM	F06YJF (DTRSM)
STRSV	DTRSV	F06PJF (DTRSV)
SLANSF	DLANSF	F06WAF (DLANSF)
STFSM	DTFSM	F06WBF (DTFSM)
SSFRK	DSFRK	F06WCF (DSFRK)

Complex Matrices		
single precision	double precision	NAG
ICAMAX	IZAMAX	F06JMF (IZAMAX)
CAXPY	ZAXPY	F06GCF (ZAXPY)
CAXPYI	ZAXPYI	F06GTF (ZAXPYI)
CCOPY	ZCOPY	F06GFF (ZCOPY)
CDOTC	ZDOTC	F06GBF (ZDOTC)
CDOTCI	ZDOTCI	F06GSF (ZDOTCI)
CDOTU	ZDOTU	F06GAF (ZDOTU)
CDOTUI	ZDOTUI	F06GRF (ZDOTUI)
CGBMV	ZGBMV	F06SBF (ZGBMV)
CGEMM	ZGEMM	F06ZAF (ZGEMM)
CGEMV	ZGEMV	F06SAF (ZGEMV)
CGERC	ZGERC	F06SNF (ZGERC)
CGERU	ZGERU	F06SMF (ZGERU)
CGTHR	ZGTHR	F06GUF (ZGTHR)
CGTHRZ	ZGTHRZ	F06GVF (ZGTHRZ)
CHBMV	ZHBMV	F06SDF (ZHBMV)
CHEMM	ZHEMM	F06ZCF (ZHEMM)
CHEMV	ZHEMV	F06SCF (ZHEMV)
CHER	ZHER	F06SPF (ZHER)
CHER2	ZHER2	F06SRF (ZHER2)
CHER2K	ZHER2K	F06ZRF (ZHER2K)
CHERK	ZHERK	F06ZPF (ZHERK)
CHPMV	ZHPMV	F06SEF (ZHPMV)
CHPR	ZHPR	F06SQF (ZHPR)
CHPR2	ZHPR2	F06SSF (ZHPR2)
CSCAL	ZSCAL	F06GDF (ZSCAL)
CSCTR	ZSCTR	F06GWF (ZSCTR)
CSSCAL	ZDSCAL	F06JDF (ZDSCAL)
CSWAP	ZSWAP	F06GGF (ZSWAP)
CSYMM	ZSYMM	F06ZTF (ZSYMM)
CSYR2K	ZSYR2K	F06ZWF (ZSYR2K)
CSYRK	ZSYRK	F06ZUF (ZSYRK)
CTBMV	ZTBMV	F06SGF (ZTBMV)
CTBSV	ZTBSV	F06SKF (ZTBSV)
CTPMV	ZTPMV	F06SHF (ZTPMV)
CTPSV	ZTPSV	F06SLF (ZTPSV)
CTRMM	ZTRMM	F06ZFF (ZTRMM)
CTRMV	ZTRMV	F06SFF (ZTRMV)
CTRSM	ZTRSM	F06ZJF (ZTRSM)
CTRSV	ZTRSV	F06SJF (ZTRSV)
CLANHF	ZLANHF	F06WNF (ZLANHF)
CHFRK	ZHFRK	F06WQF (ZHFRK)
CTFSM	ZTFSM	F06WPF (ZTFSM)

5 Auxiliary Routines Associated with Library Routine Arguments

None.

6 Routines Withdrawn or Scheduled for Withdrawal

None.

7 References

Dodson D S and Grimes R G (1982) Remark on Algorithm 539 *ACM Trans. Math. Software* **8** 403–404

Dodson D S, Grimes R G and Lewis J G (1991) Sparse extensions to the Fortran basic linear algebra subprograms *ACM Trans. Math. Software* **17** 253–263

Dongarra J J, Du Croz J J, Duff I S and Hammarling S (1990) A set of Level 3 basic linear algebra subprograms *ACM Trans. Math. Software* **16** 1–28

Dongarra J J, Du Croz J J, Hammarling S and Hanson R J (1988) An extended set of FORTRAN basic linear algebra subprograms *ACM Trans. Math. Software* **14** 1–32

Dongarra J J, Moler C B, Bunch J R and Stewart G W (1979) *LINPACK Users' Guide* SIAM, Philadelphia

Golub G H and Van Loan C F (1996) *Matrix Computations* (3rd Edition) Johns Hopkins University Press, Baltimore

Lawson C L, Hanson R J, Kincaid D R and Krogh F T (1979) Basic linear algebra supbrograms for Fortran usage *ACM Trans. Math. Software* **5** 308–325