Chapter 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.

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.* [7], Dodson *et al.* [2], Dongarra *et al.* [4] and [5]. 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.

Table 1 indicates the naming scheme for the routines in this chapter. The heading BLAS in the table indicates that routines in that category meet the specification of the BLAS, 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.

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

Table 1

The routines in this chapter do not have full routine documents, but instead are covered by some relevant background material, in Section 2.2, together with general descriptions, in Section 4, sufficient to enable their use. Descriptions of the individual routines are included in the NAG online documentation. As this chapter is concerned only with basic linear algebra operations, the routines will not normally be required by the general user. The functionality of each routine is indicated in Section 4 so that those users requiring these routines to build specialist linear algebra modules can determine which routines are of interest.

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.* [2], Dongarra *et al.* [4] and [5]).

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 [6].

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

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

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

$$R = \begin{pmatrix} c & s \\ -s & c \end{pmatrix}, \quad c = \cos\theta, \quad s = \sin\theta. \tag{1}$$

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 (SROTG/DROTG), see Lawson *et al.* [7] and Dodson and Grimes [1], computes c, s and d as

$$d = \sigma(a^2 + b^2)^{1/2},$$

$$c = \begin{cases} a/d, & d \neq 0, \\ 1, & d = 0, \end{cases} \quad s = \begin{cases} b/d, & d \neq 0, \\ 0, & d = 0, \end{cases} \tag{4}$$

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

The value z defined as

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

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

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

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

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

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

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

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 Fortran 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

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

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

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

where $\bar{a}$ denotes the complex conjugate of a. The BLAS (Lawson *et al.* [7]) 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

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

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

When s is real then the transformation

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

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 [6].

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

$$P = I - \mu u u^T, \qquad \mu u^T u = 2, \tag{12}$$

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

$$u = \begin{pmatrix} \zeta \\ z \end{pmatrix}, \qquad \zeta \text{ a scalar} \tag{13}$$

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

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

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.* [3]) 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 Fortran 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

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

and so we can call a matrix-vector product routine to form $b = A^T u$ and then call a rank-one update routine to form $(A - \mu u b^T)$. 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

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

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

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

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

The 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 parameters the 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

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

The special setting

$$\theta = 0$$

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

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

is used to flag the case where

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

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

Note. Refer to the Users' Note for your implementation to check that a routine is available.

This section lists the routines in each of the categories Level-0 (scalar), Level-1 (vector), Level-2 (matrix-vector and matrix) and Level-3 (matrix-matrix). In each case a separate sub-section is given for the routines that meet the specification of the BLAS and for the other F06 routines. For routines that meet the specification of the BLAS, the corresponding BLAS name is indicated in brackets; in single precision implementations the first of the names in the brackets is the appropriate name and in double precision implementations it is the second of the names that is appropriate.

Within each section routines are listed in alphabetic order of the fifth character in the routine name, so that corresponding real and complex routines may have adjacent entries.

3.1 The Level-0 Scalar Routines

The Level-0 routines just perform scalar operations such as generating a plane rotation.

3.1.1 The BLAS Level-0 scalar routine

F06AAF (SROTG/DROTG) generates a real plane rotation

3.1.2 The F06 Level-0 scalar routines

F06BAF generates a real plane rotation, storing the tangent
F06CAF generates a complex plane rotation, storing the tangent (real cosine)
F06CBF generates a complex plane rotation, storing the tangent (real sine)
F06BCF recovers the cosine and sine from a given real tangent
F06CCF recovers the cosine and sine from a given complex tangent (real cosine)
F06CDF recovers the cosine and sine from a given complex tangent (real sine)
F06BEF generates a real Jacobi plane rotation
F06BHF applies a real similarity rotation to a 2×2 symmetric matrix
F06CHF applies a complex similarity rotation to a 2×2 Hermitian matrix
F06BLF divides two real scalars, with an overflow flag
F06CLF divides two complex scalars, with an overflow flag
F06BMF calculates the Euclidean length of a vector following the use of routines F06FJF or F06KJF
F06BNF computes the value $(a^2 + b^2)^{1/2}$; a, b real
F06BPF computes an eigenvalue of a 2×2 real symmetric matrix

3.2 The Level-1 Vector Routines

The Level-1 routines perform operations on or between vectors, such as computing dot products and Euclidean lengths.

3.2.1 The BLAS Level-1 vector and sparse vector routines

F06EAF (SDOT/DDOT)	computes the dot product of two real vectors
F06GAF (CDOTU/ZDOTU)	computes the dot product of two complex vectors (unconjugated)
F06GBF (CDOTC/ZDOTC)	computes the dot product of two complex vectors (conjugated)
F06ECF (SAXPY/DAXPY)	adds a scalar times a vector to another real vector
F06GCF (CAXPY/ZAXPY)	adds a scalar times a vector to another complex vector
F06EDF (SSCAL/DSCAL)	multiplies a real vector by a scalar
F06GDF (CSCAL/ZSCAL)	multiplies a complex vector by a scalar
F06JDF (CSSCAL/ZDSCAL)	multiplies a complex vector by a real scalar
F06EFF (SCOPY/DCOPY)	copies a real vector
F06GFF (CCOPY/ZCOPY)	copies a complex vector
F06EGF (SSWAP/DSWAP)	swaps two real vectors
F06GGF (CSWAP/ZSWAP)	swaps two complex vectors
F06EJF (SNRM2/DNRM2)	computes the Euclidean length of a real vector
F06JJF (SCNRM2/DZNRM2)	computes the Euclidean length of a complex vector
F06EKF (SASUM/DASUM)	sums the absolute values of the elements of a real vector
F06JKF (SCASUM/DZASUM)	sums the absolute values of the elements of a complex vector
F06JLF (ISAMAX/IDAMAX)	finds the index of the element of largest absolute value of a real vector
F06JMF (ICAMAX/IZAMAX)	finds the index of the element of largest absolute value of a complex vector
F06EPF (SROT/DROT)	applies a real plane rotation
F06ERF (SDOTI/DDOTI)	computes the dot product of two real sparse vectors
F06GRF (CDOTUI/ZDOTUI)	computes the dot product of two complex sparse vectors (unconjugated)
F06GSF (CDOTCI/ZDOTCI)	computes the dot product of two complex sparse vectors (conjugated)
F06ETF (SAXPYI/DAXPYI)	adds a scalar times a sparse vector to another real sparse vector
F06GTF (CAXPYI/ZAXPYI)	adds a scalar times a sparse vector to another complex sparse vector
F06EUF (SGTHR/DGTHR)	gathers a real sparse vector
F06GUF (CGTHR/ZGTHR)	gathers a complex sparse vector
F06EVF (SGTHRZ/DGTHRZ)	gathers and sets to zero a real sparse vector
F06GVF (CGTHRZ/ZGTHRZ)	gathers and sets to zero a complex sparse vector
F06EWF (SSCTR/DSCTR)	scatters a real sparse vector
F06GWF (CSCTR/ZSCTR)	scatters a complex sparse vector
F06EXF (SROTI/DROTI)	applies a plane rotation to two real sparse vectors

3.2.2 The F06 Level-1 vector routines

F06FAF	computes the cosine of the angle between two real vectors
F06DBF	loads a scalar into each element of an integer vector
F06FBF	loads a scalar into each element of a real vector
F06HBF	loads a scalar into each element of a complex vector
F06FCF	multiplies a real vector by a diagonal matrix
F06HCF	multiplies a complex vector by a diagonal matrix

F06KCF multiplies a complex vector by a real diagonal matrix
F06FDF multiplies a real vector by a scalar, preserving the input vector
F06HDF multiplies a complex vector by a scalar, preserving the input vector
F06KDF multiplies a complex vector by a real scalar, preserving the input vector
F06DFF copies an integer vector
F06KFF copies a real vector to a complex vector
F06FGF negates a real vector
F06HGF negates a complex vector
F06FJF updates the Euclidean length of a real vector in scaled form
F06KJF updates the Euclidean length of a complex vector in scaled form
F06FKF finds the weighted Euclidean length of a real vector
F06FLF finds the elements of largest and smallest absolute value of a real vector
F06KLF finds the last non-negligible element of a real vector
F06FPF applies a real symmetric plane rotation
F06HPF applies a complex plane rotation
F06KPF applies a real plane rotation to two complex vectors
F06FQF generates a sequence of real plane rotations
F06HQF generates a sequence of complex plane rotations
F06FRF generates a real elementary reflection (NAG style)
F06HRF generates a complex elementary reflection
F06FSF generates a real elementary reflection (LINPACK style)
F06FTF applies a real elementary reflection (NAG style)
F06HTF applies a complex elementary reflection
F06FUF applies a real elementary reflection (LINPACK style)

3.3 The Level-2 Matrix-vector and Matrix Routines

The Level-2 routines perform matrix-vector and matrix operations, such as forming the product between a matrix and a vector, computing Frobenius norms and applying a sequence of plane rotations.

3.3.1 The BLAS Level-2 matrix-vector routines

F06PAF (SGEMV/DGEMV) computes a matrix-vector product; real general matrix
F06SAF (CGEMV/ZGEMV) computes a matrix-vector product; complex general matrix
F06PBF (SGBMV/DGBMV) computes a matrix-vector product; real general band matrix
F06SBF (CGBMV/ZGBMV) computes a matrix-vector product; complex general band matrix
F06PCF (SSYMV/DSYMV) computes a matrix-vector product; real symmetric matrix
F06SCF (CHEMV/ZHEMV) computes a matrix-vector product; complex Hermitian matrix
F06PDF (SSBMV/DSBMV) computes a matrix-vector product; real symmetric band matrix
F06SDF (CHBMV/ZHBMV) computes a matrix-vector product; complex Hermitian band matrix
F06PEF (SSPMV/DSPMV) computes a matrix-vector product; real symmetric packed matrix
F06SEF (CHPMV/ZHPMV) computes a matrix-vector product; complex Hermitian packed matrix
F06PFF (STRMV/DTRMV) computes a matrix-vector product; real triangular matrix
F06SFF (CTRMV/ZTRMV) computes a matrix-vector product; complex triangular matrix
F06PGF (STBMV/DTBMV) computes a matrix-vector product; real triangular band matrix
F06SGF (CTBMV/ZTBMV) computes a matrix-vector product; complex triangular band matrix
F06PHF (STPMV/DTPMV) computes a matrix-vector product; real triangular packed matrix

F06SHF (CTPMV/ZTPMV)	computes a matrix-vector product; complex triangular packed matrix
F06PJF (STRSV/DTRSV)	solves a system of equations; real triangular coefficient matrix
F06SJF (CTRSV/ZTRSV)	solves a system of equations; complex triangular coefficient matrix
F06PKF (STBSV/DTBSV)	solves a system of equations; real triangular band coefficient matrix
F06SKF (CTBSV/ZTBSV)	solves a system of equations; complex triangular band coefficient matrix
F06PLF (STPSV/DTPSV)	solves a system of equations; real triangular packed coefficient matrix
F06SLF (CTPSV/ZTPSV)	solves a system of equations; complex triangular packed coefficient matrix
F06PMF (SGER/DGER)	performs a rank-one update; real general matrix
F06SMF (CGERU/ZGERU)	performs a rank-one update; complex general matrix (unconjugated vector)
F06SNF (CGERC/ZGERC)	performs a rank-one update; complex general matrix (conjugated vector)
F06PPF (SSYR/DSYR)	performs a rank-one update; real symmetric matrix
F06SPF (CHER/ZHER)	performs a rank-one update; complex Hermitian matrix
F06PQF (SSPR/DSPR)	performs a rank-one update; real symmetric packed matrix
F06SQF (CHPR/ZHPR)	performs a rank-one update; complex Hermitian packed matrix
F06PRF (SSYR2/DSYR2)	performs a rank-two update; real symmetric matrix
F06SRF (CHER2/ZHER2)	performs a rank-two update; complex Hermitian matrix
F06PSF (SSPR2/DSPR2)	performs a rank-two update; real symmetric packed matrix
F06SSF (CHPR2/ZHPR2)	performs a rank-two update; complex Hermitian packed matrix

3.3.2 The Level-2 matrix routines

F06QFF	copies a real general or trapezoidal matrix
F06TFF	copies a complex general or trapezoidal matrix
F06QHF	loads a scalar into each element of a real general or trapezoidal matrix; a different scalar may be loaded into the diagonal elements
F06THF	loads a scalar into each element of a complex general or trapezoidal matrix; a different scalar may be loaded into the diagonal elements
F06QJF	applies a sequence of permutation matrices, represented by an integer array, to a real general matrix
F06VJF	applies a sequence of permutation matrices, represented by an integer array, to a complex general matrix
F06QKF	applies a sequence of permutation matrices, represented by a real array, to a real general matrix
F06VKF	applies a sequence of permutation matrices, represented by a real array, to a complex general matrix
F06QMF	applies a sequence of plane rotations, as a similarity transformation, to a real symmetric matrix
F06TMF	applies a sequence of plane rotations, as a similarity transformation, to a complex Hermitian matrix
F06QPF	applies a rank-one update to a real upper triangular matrix, maintaining upper triangular form
F06TPF	applies a rank-one update to a complex upper triangular matrix, maintaining upper triangular form
F06QQF	performs a QR factorization of a real upper triangular matrix augmented by an additional full row

F06TQF performs a QR factorization of a complex upper triangular matrix augmented by an additional full row

F06QRF applies a sequence of plane rotations, from either the left or the right, to reduce a real upper Hessenberg matrix to upper triangular form

F06TRF applies a sequence of plane rotations, from either the left or the right, to reduce a complex upper Hessenberg matrix to upper triangular form

F06QSF applies a sequence of plane rotations, from either the left or the right, to reduce a real upper spiked matrix to upper triangular form

F06TSF applies a sequence of plane rotations, from either the left or the right, to reduce a complex upper spiked matrix to upper triangular form

F06QTF applies a given sequence of plane rotations, from either the left or the right, to a real upper triangular matrix and reduces the resulting matrix back to upper triangular form by applying plane rotations from the other side

F06TTF applies a given sequence of plane rotations, from either the left or the right, to a complex upper triangular matrix and reduces the resulting matrix back to upper triangular form by applying plane rotations from the other side

F06QVF applies a given sequence of plane rotations, from either the left or the right, to a real upper triangular matrix to give an upper Hessenberg matrix

F06TVF applies a given sequence of plane rotations, from either the left or the right, to a complex upper triangular matrix to give an upper Hessenberg matrix

F06QWF applies a given sequence of plane rotations, from either the left or the right, to a real upper triangular matrix to give an upper spiked matrix

F06TWF applies a given sequence of plane rotations, from either the left or the right, to a complex upper triangular matrix to give an upper spiked matrix

F06QXF applies a given sequence of plane rotations, from either the left or the right, to a real general matrix

F06TXF applies a given sequence of plane rotations with real cosines, from either the left or the right, to a complex general matrix

F06TYF applies a given sequence of plane rotations with real sines, from either the left or the right, to a complex general matrix

F06VXF applies a given sequence of real plane rotations, from either the left or the right, to a complex general matrix

F06RAF computes a norm, or the element of largest absolute value of a real general matrix

F06UAF computes a norm, or the element of largest absolute value of a complex general matrix

F06RBF computes a norm, or the element of largest absolute value of a real band matrix

F06UBF computes a norm, or the element of largest absolute value of a complex band matrix

F06RCF computes a norm, or the element of largest absolute value of a real symmetric matrix

F06UCF computes a norm, or the element of largest absolute value of a complex Hermitian matrix

F06RDF computes a norm, or the element of largest absolute value of a real symmetric matrix stored in packed form

F06UDF computes a norm, or the element of largest absolute value of a complex Hermitian matrix stored in packed form

F06REF computes a norm, or the element of largest absolute value of a real symmetric band matrix

F06UEF computes a norm, or the element of largest absolute value of a complex Hermitian band matrix

F06RJF computes a norm, or the element of largest absolute value of a real general trapezoidal matrix

F06UJF computes a norm, or the element of largest absolute value of a complex general trapezoidal matrix

F06RKF computes a norm, or the element of largest absolute value of a real triangular matrix stored in packed form

F06UKF computes a norm, or the element of largest absolute value of a complex triangular matrix stored in packed form

F06RLF computes a norm, or the element of largest absolute value of a real triangular band matrix

F06ULF computes a norm, or the element of largest absolute value of a complex triangular band matrix

F06RMF computes a norm, or the element of largest absolute value of a real Hessenberg matrix

F06UMF computes a norm, or the element of largest absolute value of a complex Hessenberg matrix

F06UFF computes a norm, or the element of largest absolute value of a complex symmetric matrix

F06UGF computes a norm, or the element of largest absolute value of a complex symmetric matrix stored in packed form

F06UHF computes a norm, or the element of largest absolute value of a complex symmetric band matrix

3.4 The Level-3 Matrix-matrix Routines

The Level-3 routines perform matrix-matrix operations, such as forming the product of two matrices.

3.4.1 The BLAS Level-3 matrix-matrix routines

F06YAF (SGEMM/DGEMM) computes a matrix-matrix product; two real rectangular matrices

F06ZAF (CGEMM/ZGEMM) computes a matrix-matrix product; two complex rectangular matrices

F06YCF (SSYMM/DSYMM) computes a matrix-matrix product; one real symmetric matrix, one real rectangular matrix

F06ZCF (CHEMM/ZHEMM) computes a matrix-matrix product; one complex Hermitian matrix, one complex rectangular matrix

F06YFF (STRMM/DTRMM) computes a matrix-matrix product; one real triangular matrix, one real rectangular matrix

F06ZFF (CTRMM/ZTRMM) computes a marix-matrix product; one complex triangular matrix, one complex rectangular matrix

F06YJF (STRSM/DTRSM) solves a system of equations with multiple right-hand sides, real triangular coefficient matrix

F06ZJF (CTRSM/ZTRSM) solves a system of equations with multiple right-hand sides, complex triangular coefficient matrix

F06YPF (SSYRK/DSYRK) performs a rank-k update of a real symmetric matrix

F06ZPF (CHERK/ZHERK) performs a rank-k update of a complex hermitian matrix

F06YRF (SSYR2K/DSYR2K) performs a rank-$2k$ update of a real symmetric matrix

F06ZRF (CHER2K/ZHER2K) performs a rank-$2k$ update of a complex Hermitian matrix

F06ZTF (CSYMM/ZSYMM) computes a matrix-matrix product: one complex symmetric matrix, one complex rectangular matrix

F06ZUF (CSYRK/ZSYRK) performs a rank-k update of a complex symmetric matrix

F06ZWF (CSYR2K/ZSYR2K) performs a rank-$2k$ update of a complex symmetric matrix

4 Description of the F06 Routines

In this section we describe the purpose of each routine and give information on the parameter lists, where appropriate indicating their general nature. Usually the association between the routine arguments and the mathematical variables is obvious and in such cases a description of the argument is omitted.

Within each section, the parameter lists for all routines are presented, followed by the purpose of the routines and information on the parameter lists.

For those routines that meet the specification of the BLAS, the parameter lists indicate the single precision BLAS name, but this should be substituted by the double precision BLAS name in double precision implementations (see Sections 3.1–3.4).

Within each section routines are listed in alphabetic order of the fifth character in the routine name, so that corresponding real and complex routines may have adjacent entries.

4.1 The Level-0 Scalar Routines

The scalar routines have no array arguments.

4.1.1 The BLAS Level-0 scalar routine

```
SUBROUTINE            F06AAF    ( A,B,C,S )
ENTRY                 srotg     ( A,B,C,S )
real                              A,B,C,S
```

F06AAF generates the parameters c and s of a Givens rotation as defined by equations (4) and (5), from given a and b. On exit, A is overwritten by d and B is overwritten by z.

4.1.2 The F06 scalar routines

```
SUBROUTINE            F06BAF    ( A,B,C,S )
real                              A,B,C,S

SUBROUTINE            F06CAF    ( A,B,C,S )
complex                           A,B,  S
real                                 C

SUBROUTINE            F06CBF    ( A,B,C,S )
complex                           A,B,C
real                                   S

SUBROUTINE            F06BCF    (   T,C,S )
real                              T,C,S

SUBROUTINE            F06CCF    (   T,C,S )
complex                           T,  S
real                                 C

SUBROUTINE            F06CDF    (   T,C,S )
complex                           T,C
real                                   S

SUBROUTINE            F06BEF    ( JOB,X,Y,Z,C,S )
CHARACTER*1                       JOB
real                                 X,Y,Z,C,S

SUBROUTINE            F06BHF    (     X,Y,Z,C,S )
real                              X,Y,Z,C,S

SUBROUTINE            F06CHF    (     X,Y,Z,C,S )
complex                           X,Y,Z,  S
real                                       C

real FUNCTION         F06BLF    ( A,B,FAIL )
real                              A,B
LOGICAL                              FAIL

complex FUNCTION      F06CLF    ( A,B,FAIL )
complex                           A,B
LOGICAL                              FAIL

real FUNCTION         F06BMF    ( SCALE,SSQ )
real                              SCALE,SSQ

real FUNCTION         F06BNF    ( A,B )
real                              A,B

real FUNCTION         F06BPF    ( X,Y,Z )
real                              X,Y,Z
```

F06BAF, **F06CAF** and **F06CBF** generate the parameters c and s of a Givens rotation as defined by equations (6), (7) and their complex equivalents, from given a and b. On exit, A is overwritten by d and B is overwritten by t.

F06BCF, **F06CCF** and **F06CDF** recover the parameters c and s of a plane rotation from a given value of t.

F06BEF generates the parameters c and s of a Jacobi rotation from given x, y and z (see equation (8)). The input parameter JOB controls the choice of rotation as follows:

JOB = 'B', then $c \geq 1/\sqrt{2}$,
JOB = 'S', then $0 \leq c \leq 1/\sqrt{2}$,
JOB = 'M', then $|a| \geq |b|$.

On exit, a and b are overwritten on X and Z, and t is overwritten on Y.

F06BHF and **F06CHF** apply a similarity plane rotation to a two by two symmetric or Hermitian matrix defined by x, y and z. X, Y and Z are overwritten by the transformed elements.

F06BLF and **F06CLF** return the value a/b, unless overflow would occur. If overflow would occur then the value zero is returned when $a = 0$ and a value big, defined as follows, is returned otherwise. For F06BLF big is defined as

$$big = flmax.\text{sign}(a/b)$$

and for F06CLF big is defined as

$$big = flmax.(\text{sign}(\text{Re}(a/b)) + i.\text{sign}(\text{Im}(a/b))),$$

where $flmax$ is the reciprocal of the value returned by X02AMF and sign(a/b) is taken as sign(a) when $b = 0$. The argument FAIL is returned as *false* when overflow would not occur and is returned as *true* otherwise.

F06BMF returns the value $scale.\sqrt{sumsq}$. This routine is intended to be used following either of the routines F06FJF or F06KJF.

F06BNF returns the value $(a^2 + b^2)^{1/2}$, for given a and b.

F06BPF returns an eigenvalue of a two by two symmetric matrix. The eigenvalue λ is given by

$$\lambda = z - y/(f + \text{sign}(f).(1 + f^2)^{1/2}), \quad \text{where } f = (x - z)/(2y).$$

When $y = 0$ then $\lambda = z$.

4.2 The Level-1 Vector Routines

The vector routines all have one or more one-dimensional arrays as arguments, each representing a vector.

In the non-sparse case the length of each vector, n, is represented by the argument N, and the routines may be called with non-positive values of N, in which case the routine returns immediately except for the functions, which set the function value to zero before returning.

In addition to the argument N, each array argument is also associated with an **increment** argument that immediately follows the array argument, and whose name consists of the three characters INC, followed by the name of the array. For example, a vector x will be represented by the two arguments X, INCX. The increment argument is the spacing (stride) in the array for which the elements of the vector occur. For instance, if INCX=2, then the elements of x are in locations X(1),X(3),...,X(2∗N−1) of the array X and the intermediate locations X(2),X(4),...,X(2∗N−2) are not referenced.

Thus when INCX >0, the vector element x_i is in the array element X(1+$(i-1)$∗INCX). When INCX ≤ 0 the elements are stored in the reverse order so that the vector element x_i is in the array element X$(1 - (n - i) *$ INCX) and hence, in particular, the element x_n is in X(1). The declared length of the array X in the calling (sub)program must be at least $(1 + (\text{N} - 1) * |\text{INCX}|)$.

Non-positive increments are permitted only for those routines that have more than one array argument. While zero increments are formally permitted for such routines, their use in Chapter F06 is strongly

discouraged since the effect may be implementation dependent. There will usually be an alternative routine, with a simplified parameter list, to achieve the required purpose.

In the sparse case the routines are all concerned with operations on two sparse n element vectors x and y. The vector x is stored in a dense (compressed) one-dimensional array X containing only the interesting (usually non-zero) elements of x, while y is stored in full uncompressed form in an n element array Y. The vector x is represented by the three arguments NZ, X and INDX, where NZ is the number of interesting elements of x and INDX is a one-dimensional (index) array such that

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

The vector y is represented only by the argument Y; no increment arguments are included.

Non-positive values of NZ are permitted, in which case the routine returns immediately except for functions, which set the function 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.

4.2.1 The BLAS Level-1 vector routines

```
real FUNCTION          F06EAF      ( N,       X,INCX,Y,INCY )
real                   sdot
ENTRY                  sdot        ( N,       X,INCX,Y,INCY )
INTEGER                              N,         INCX,INCY
real                                            X(*),  Y(*)

complex FUNCTION       F06GAF      ( N,       X,INCX,Y,INCY )
complex                cdotu
ENTRY                  cdotu       ( N,       X,INCX,Y,INCY )
INTEGER                              N,         INCX,  INCY
complex                                         X(*),  Y(*)

complex FUNCTION       F06GBF      ( N,       X,INCX,Y,INCY )
complex                cdotc
ENTRY                  cdotc       ( N,       X,INCX,Y,INCY )
INTEGER                              N,         INCX,  INCY
complex                                         X(*),  Y(*)

SUBROUTINE             F06ECF      ( N,ALPHA, X,INCX,Y,INCY )
ENTRY                  saxpy       ( N,ALPHA, X,INCX,Y,INCY )
INTEGER                              N,         INCX,  INCY
real                                   ALPHA, X(*),  Y(*)

SUBROUTINE             F06GCF      ( N,ALPHA, X,INCX,Y,INCY )
ENTRY                  caxpy       ( N,ALPHA, X,INCX,Y,INCY )
INTEGER                              N,         INCX,  INCY
complex                                ALPHA, X(*),  Y(*)

SUBROUTINE             F06EDF      ( N,ALPHA, X,INCX )
ENTRY                  sscal       ( N,ALPHA, X,INCX )
INTEGER                              N,         INCX
real                                   ALPHA, X(*)

SUBROUTINE             F06GDF      ( N,ALPHA, X,INCX )
ENTRY                  cscal       ( N,ALPHA, X,INCX )
INTEGER                              N,         INCX
complex                                ALPHA, X(*)

SUBROUTINE             F06JDF      ( N,ALPHA, X,INCX )
ENTRY                  csscal      ( N,ALPHA, X,INCX )
INTEGER                              N,         INCX
real                                   ALPHA
complex                                         X(*)
```

```
SUBROUTINE            F06EFF      ( N,      X,INCX,Y,INCY )
ENTRY                 scopy       ( N,      X,INCX,Y,INCY )
INTEGER                             N,        INCX,  INCY
real                                          X(*),  Y(*)

SUBROUTINE            F06GFF      ( N,      X,INCX,Y,INCY )
ENTRY                 ccopy       ( N,      X,INCX,Y,INCY )
INTEGER                             N,        INCX,  INCY
complex                                       X(*),  Y(*)

SUBROUTINE            F06EGF      ( N,      X,INCX,Y,INCY )
ENTRY                 sswap       ( N,      X,INCX,Y,INCY )
INTEGER                             N,        INCX,  INCY
real                                          X(*),  Y(*)

SUBROUTINE            F06GGF      ( N,      X,INCX,Y,INCY )
ENTRY                 cswap       ( N,      X,INCX,Y,INCY )
INTEGER                             N,        INCX,  INCY
complex                                       X(*),  Y(*)

real FUNCTION         F06EJF      ( N,      X,INCX )
real                  snrm2
ENTRY                 snrm2       ( N,      X,INCX )
INTEGER                             N,        INCX
real                                          X(*)

real FUNCTION         F06JJF      ( N,      X,INCX )
real                  scnrm2
ENTRY                 scnrm2      ( N,      X,INCX )
INTEGER                             N,        INCX
complex                                       X(*)

real FUNCTION         F06EKF      ( N,      X,INCX )
real                  sasum
ENTRY                 sasum       ( N,      X,INCX )
INTEGER                             N,        INCX
real                                          X(*)

real FUNCTION         F06JKF      ( N,      X,INCX )
real                  scasum
ENTRY                 scasum      ( N,      X,INCX )
INTEGER                             N,        INCX
complex                                       X(*)

INTEGER FUNCTION      F06JLF      ( N,      X,INCX )
INTEGER               isamax
ENTRY                 isamax      ( N,      X,INCX )
INTEGER                             N,        INCX
real                                          X(*)

INTEGER FUNCTION      F06JMF      ( N,      X,INCX )
INTEGER               icamax
ENTRY                 icamax      ( N,      X,INCX )
INTEGER                             N,        INCX
complex                                       X(*)

SUBROUTINE            F06EPF      ( N,      X,INCX,Y,INCY,C,S )
ENTRY                 srot        ( N,      X,INCX,Y,INCY,C,S )
INTEGER                             N,        INCX,  INCY
real                                          X(*),  Y(*),  C,S
```

```
real FUNCTION          F06ERF      ( NZ,      X,INDX,Y )
real                   sdoti
ENTRY                  sdoti       ( NZ,      X,INDX,Y )
INTEGER                            NZ,        INDX(*)
real                                          X(*),  Y(*)

complex FUNCTION       F06GRF      ( NZ,      X,INDX,Y )
complex                cdotui
ENTRY                  cdotui      ( NZ,      X,INDX,Y )
INTEGER                            NZ,        INDX(*)
complex                                       X(*),  Y(*)

complex FUNCTION       F06GSF      ( NZ,      X,INDX,Y )
complex                cdotci
ENTRY                  cdotci      ( NZ,      X,INDX,Y )
INTEGER                            NZ,        INDX(*)
complex                                       X(*),  Y(*)

SUBROUTINE             F06ETF      ( NZ,ALPHA,X,INDX,Y )
ENTRY                  saxpyi      ( NZ,ALPHA,X,INDX,Y )
INTEGER                            NZ,        INDX(*)
real                                  ALPHA,X(*),  Y(*)

SUBROUTINE             F06GTF      ( NZ,ALPHA,X,INDX,Y )
ENTRY                  caxpyi      ( NZ,ALPHA,X,INDX,Y )
INTEGER                            NZ,        INDX(*)
complex                               ALPHA,X(*),  Y(*)

SUBROUTINE             F06EUF      ( NZ, Y,   X,INDX )
ENTRY                  sgthr       ( NZ, Y,   X,INDX )
INTEGER                            NZ,        INDX(*)
real                                   Y(*),X(*)

SUBROUTINE             F06GUF      ( NZ, Y,   X,INDX )
ENTRY                  cgthr       ( NZ, Y,   X,INDX )
INTEGER                            NZ,        INDX(*)
complex                                Y(*),X(*)

SUBROUTINE             F06EVF      ( NZ, Y,   X,INDX )
ENTRY                  sgthrz      ( NZ, Y,   X,INDX )
INTEGER                            NZ,        INDX(*)
real                                   Y(*),X(*)

SUBROUTINE             F06GVF      ( NZ, Y,   X,INDX )
ENTRY                  cgthrz      ( NZ, Y,   X,INDX )
INTEGER                            NZ,        INDX(*)
complex                                Y(*),X(*)

SUBROUTINE             F06EWF      ( NZ,      X,INDX,Y )
ENTRY                  ssctr       ( NZ,      X,INDX,Y )
INTEGER                            NZ,        INDX(*)
real                                          X(*),  Y(*)

SUBROUTINE             F06GWF      ( NZ,      X,INDX,Y )
ENTRY                  csctr       ( NZ,      X,INDX,Y )
INTEGER                            NZ,        INDX(*)
complex                                       X(*),  Y(*)

SUBROUTINE             F06EXF      ( NZ,      X,INDX,Y, C,S )
ENTRY                  sroti       ( NZ,      X,INDX,Y, C,S )
INTEGER                            NZ,        INDX(*)
real                                          X(*),Y(*),C,S
```

F06EAF, **F06GAF**, **F06ERF** and **F06GRF** return the dot product $x^T y$.

F06GBF and **F06GSF** return the dot product $x^H y$, where x^H denotes the complex conjugate of x^T.

F06ECF, **F06GCF**, **F06ETF** and **F06GTF** perform the operation

$$y \leftarrow \alpha x + y,$$

often called an **axpy** operation.

F06EDF, **F06GDF** and **F06JDF** perform the operation

$$x \leftarrow \alpha x.$$

F06EFF, **F06GFF**, **F06EWF** and **F06GWF** perform the operation

$$y \leftarrow x.$$

F06EGF and **F06GGF** perform the operation

$$x \Leftrightarrow y,$$

that is x and y are swapped.

F06EJF and **F06JJF** return the value $\|x\|_2$ defined by

$$\|x\|_2 = \left(\sum_{i=1}^{n} |x_i|^2 \right)^{1/2}.$$

F06EKF returns the value $\|x\|_1$ defined by

$$\|x\|_1 = \sum_{i=1}^{n} |x_i|.$$

F06JKF returns the value *asum* defined by

$$asum = \sum_{i=1}^{n} (|\operatorname{Re}(x_i)| + |\operatorname{Im}(x_i)|).$$

F06JLF returns the first index j such that

$$|x_j| = \max_i |x_i|.$$

F06JMF returns the first index j such that

$$|\operatorname{Re}(x_j)| + |\operatorname{Im}(x_j)| = \max_i (|\operatorname{Re}(x_i)| + |\operatorname{Im}(x_i)|).$$

F06EPF and **F06EXF** performs the plane rotation

$$\begin{pmatrix} x^T \\ y^T \end{pmatrix} \leftarrow \begin{pmatrix} c & s \\ -s & c \end{pmatrix} \begin{pmatrix} x^T \\ y^T \end{pmatrix}.$$

F06EUF and **F06GUF** perform the operation

$$x \leftarrow y.$$

F06EVF and **F06GVF** perform the operations

$$\begin{aligned} x &\leftarrow y \\ y &\leftarrow 0. \end{aligned}$$

4.2.2 The F06 Level-1 vector routines

```
real FUNCTION          F06FAF    ( N,J,TOLX,X,  INCX,TOLY,Y,INCY )
INTEGER                            N,J,         INCX,       INCY
real                                   TOLX,X(*),    TOLY,Y(*)

SUBROUTINE             F06DBF    ( N,CONST, X,  INCX )
INTEGER                            N,           INCX
INTEGER                              CONST, X(*)

SUBROUTINE             F06FBF    ( N,CONST, X,  INCX )
INTEGER                            N,           INCX
real                                 CONST, X(*)

SUBROUTINE             F06HBF    ( N,CONST, X,  INCX )
INTEGER                            N,           INCX
complex                              CONST, X(*)

SUBROUTINE             F06FCF    ( N,D,INCD,X,  INCX )
INTEGER                            N,  INCD,    INCX
real                                 D(*),  X(*)

SUBROUTINE             F06HCF    ( N,D,INCD,X,  INCX)
INTEGER                            N,  INCD,    INCX
complex                              D(*),  X(*)

SUBROUTINE             F06KCF    ( N,D,INCD,X,  INCX )
INTEGER                            N,  INCD,    INCX
real                                 D(*)
complex                                     X(*)

SUBROUTINE             F06FDF    ( N,ALPHA, X,  INCX,Y,INCY )
INTEGER                            N,           INCX,  INCY
real                                 ALPHA, X(*),    Y(*)

SUBROUTINE             F06HDF    ( N,ALPHA, X,  INCX,Y,INCY )
INTEGER                            N,           INCX,  INCY
complex                              ALPHA, X(*),    Y(*)

SUBROUTINE             F06KDF    ( N,ALPHA, X,  INCX,Y,INCY )
INTEGER                            N,           INCX,  INCY
real                                 ALPHA
complex                                     X(*),    Y(*)

SUBROUTINE             F06DFF    ( N,       X,  INCX,Y,INCY )
INTEGER                            N,           INCX,  INCY
INTEGER                                     X(*),    Y(*)

SUBROUTINE             F06KFF    ( N,       X,  INCX,Y,INCY )
INTEGER                            N,           INCX,  INCY
real                                        X(*)
complex                                              Y(*)

SUBROUTINE             F06FGF    ( N,       X,  INCX )
INTEGER                            N,           INCX
real                                        X(*)

SUBROUTINE             F06HGF    ( N,       X,  INCX )
INTEGER                            N,           INCX
complex                                     X(*)
```

```
SUBROUTINE            F06FJF    ( N,       X,  INCX,SCALE,SUMSQ )
INTEGER                           N,           INCX
real                                      X(*),    SCALE,SUMSQ

SUBROUTINE            F06KJF    ( N,       X,  INCX,SCALE,SUMSQ )
INTEGER                           N,           INCX
complex                                   X(*)
real                                               SCALE,SUMSQ

real FUNCTION         F06FKF    ( N,D,INCD,X,  INCX )
INTEGER                           N,  INCD,    INCX
real                                 D(*),  X(*)

SUBROUTINE            F06FLF    ( N,       X,  INCX,XMAX,XMIN )
INTEGER                           N,           INCX
real                                      X(*),    XMAX,XMIN

INTEGER FUNCTION      F06KLF    ( N,       X,  INCX,TOL )
INTEGER                           N,           INCX
real                                      X(*),    TOL

SUBROUTINE            F06FPF    ( N,       X,  INCX,Y,INCY,C,S )
INTEGER                           N,           INCX,  INCY
real                                      X(*),    Y(*),  C,S

SUBROUTINE            F06HPF    ( N,       X,  INCX,Y,INCY,C,S )
INTEGER                           N,           INCX,  INCY
complex                                   X(*),    Y(*),  C,S

SUBROUTINE            F06KPF    ( N,       X,  INCX,Y,INCY,C,S )
INTEGER                           N,           INCX,  INCY
complex                                   X(*),    Y(*)
real                                                     C,S

SUBROUTINE            F06FQF    ( PIVOT,DIRECT,N,ALPHA,X,INCX,C,   S )
CHARACTER*1                       PIVOT,DIRECT
INTEGER                                     N,        INCX
real                                          ALPHA,X(*),  C(*),S(*)

SUBROUTINE            F06HQF    ( PIVOT,DIRECT,N,ALPHA,X,INCX,C,   S )
CHARACTER*1                       PIVOT,DIRECT
INTEGER                                     N,        INCX
complex                                       ALPHA,X(*),       S(*)
real                                                        C(*)

SUBROUTINE            F06FRF    ( N,ALPHA,X,INCX,TOL,ZETA )
INTEGER                           N,        INCX
real                                 ALPHA,X(*),  TOL,ZETA

SUBROUTINE            F06HRF    ( N,ALPHA,X,INCX,TOL,THETA )
INTEGER                           N,        INCX
complex                              ALPHA,X(*),      THETA
real                                           TOL

SUBROUTINE            F06FSF    ( N,ALPHA,X,INCX,TOL,Z1 )
INTEGER                           N,        INCX
real                                 ALPHA,X(*),  TOL,Z1

SUBROUTINE            F06FTF    ( N,DELTA,Y,INCY,ZETA, Z,INCZ )
INTEGER                           N,        INCY,        INCZ
real                                 DELTA,Y(*),  ZETA, Z(*)

SUBROUTINE            F06HTF    ( N,DELTA,Y,INCY,THETA,Z,INCZ )
INTEGER                           N,        INCY,        INCZ
complex                              DELTA,Y(*),  THETA,Z(*)
```

```
SUBROUTINE            F06FUF    ( N,Z,INCZ,ZETA,DELTA,Y,INCY )
INTEGER                           N,  INCZ,            INCY
real                                Z(*),  ZETA,DELTA,Y(*)
```

F06FAF returns the value of the cosine of the angle between the vectors x and y, defined as

$$\text{F06FAF} = \frac{x^T y}{\|x\|_2 \|y\|_2},$$

where $\|x\|_2 = \sqrt{x^T x}$. If the input argument J is such that $1 \leq \text{J} \leq \text{N}$ then y is taken as

$$y = e_{\text{J}}, 1 \leq \text{J} \leq \text{N},$$

where e_{J} is the Jth column of the unit matrix and in this case Y is not referenced. If $\|x\|_2 \leq$ TOLX then F06FAF is returned as 2.0 and if $\|x\|_2 \leq$ TOLY then F06FAF is returned as -2.0, otherwise F06FAF is returned in the range $[-1.0, 1.0]$. If either TOLX or TOLY are negative then zero is used in place of the respective tolerance.

F06DBF, **F06FBF** and **F06HBF** perform the operation

$$x \leftarrow \alpha e,$$

where e is the vector $e^T = (11 \ldots 1)$.

F06FCF, **F06HCF** and **F06KCF** perform the operation

$$x \leftarrow Dx,$$

where D is a diagonal matrix, $D = \text{diag}(d_i)$.

F06FDF, **F06HDF** and **F06KDF** perform the operation

$$y \leftarrow \alpha x.$$

F06DFF and **F06KFF** perform the operation

$$y \leftarrow x.$$

F06FGF and **F06HGF** perform the operation

$$x \leftarrow -x.$$

F06FJF and **F06KJF** return the values *scl* and *ssq* given by

$$scl^2.ssq = scale^2.sumsq + \|x\|_2,$$

where for F06FJF,

$$\|x\|_2^2 = x^T x = x_1^2 + x_2^2 + \ldots + x_n^2$$

and for F06KJF,

$$\|x\|_2^2 = x^H x = |x_1|^2 + |x_2|^2 + \ldots + |x_n|^2.$$

The values of *scl* and *ssq* are overwritten on *scale* and *sumsq*, and either of these routines can be followed by routine F06BMF to compute the value $scale.\sqrt{sumsq}$. These routines are intended for the safe computation of the Euclidean lengths of vectors and matrices. Before entry, *scale* and *sumsq* are assumed to satisfy

$$0 \leq scale, \quad 1 \leq sumsq.$$

On exit from F06FJF, *scl* and *ssq* will then satisfy,

$$scl = \max_i(scale, |x_i|), \quad 1 \leq ssq \leq sumsq + n$$

and from F06KJF,

$$scl = \max_i(scale, |\,\text{Re}(x_i)|, |\,\text{Im}(x_i)|), \quad 1 \leq ssq \leq sumsq + 2n.$$

F06FKF returns the weighted Euclidean length $\|x_D\|$ defined as

$$\|x_D\| = \|D^{1/2}x\|_2 = \left(\sum_{i=1}^{n} d_i x_i^2\right)^{1/2}$$

where D is the diagonal matrix $D = \mathrm{diag}(d_i)$. The elements of D must satisfy $d_i \geq 0$.

F06FLF returns the values $xmax$ and $xmin$ given by

$$xmax = \max_i |x_i|, \quad xmin = \min_i |x_i|.$$

F06KLF returns the value $(k-1)$, where k is the smallest integer for which

$$|x_k| \leq tol.\max(|x_1|, |x_2|, \ldots, |x_{k-1}|).$$

If no such k exists then F06KLF returns the value n. If tol is less than zero on entry, then the value eps, where eps is the ***machine precision***, is used in place of tol. Note that tol is unchanged on exit. Note also that k is the index of the first negligible element in x and that $k = 1$ only if $x_1 = 0$.

F06FPF performs the symmetric plane rotation

$$\begin{pmatrix} x^T \\ y^T \end{pmatrix} \leftarrow \begin{pmatrix} c & s \\ s & -c \end{pmatrix} \begin{pmatrix} x^T \\ y^T \end{pmatrix}.$$

F06HPF performs the plane rotation

$$\begin{pmatrix} x^T \\ y^T \end{pmatrix} \leftarrow \begin{pmatrix} c & s \\ -\bar{s} & \bar{c} \end{pmatrix} \begin{pmatrix} x^T \\ y^T \end{pmatrix}.$$

Note that this differs slightly from the form given in equation (9).

F06KPF performs the plane rotation

$$\begin{pmatrix} x^T \\ y^T \end{pmatrix} \leftarrow \begin{pmatrix} c & s \\ -s & c \end{pmatrix} \begin{pmatrix} x^T \\ y^T \end{pmatrix}.$$

where x and y are complex, but c and s are real.

F06FQF and **F06HQF** generate the parameters of a sequence of plane rotations. Denoting the product of the plane rotation matrices by P, the matrix P is such that

$$\begin{pmatrix} \alpha \\ 0 \end{pmatrix} \leftarrow P \begin{pmatrix} \alpha \\ x \end{pmatrix},$$

when PIVOT = 'F' and DIRECT = 'F', or when PIVOT = 'V' and DIRECT = 'B', and

$$\begin{pmatrix} 0 \\ \alpha \end{pmatrix} \leftarrow P \begin{pmatrix} x \\ \alpha \end{pmatrix},$$

when PIVOT = 'F' and DIRECT = 'B', or when PIVOT = 'V' and DIRECT = 'F'.

When PIVOT = 'F' (Fixed pivot) and DIRECT = 'F' (Forward sequence) then P is given as the sequence

$$P = P_n, P_{n-1}, \ldots, P_1,$$

where P_k is a plane rotation matrix for the $(1, k+1)$ plane designed to annihilate the kth element of x.

When PIVOT = 'V' (Variable pivot) and DIRECT = 'B' or 'b' (Backward sequence) then P is given as the sequence

$$P = P_1, P_2, \ldots, P_n,$$

where P_k is a plane rotation matrix for the $(k, k+1)$ plane designed to annihilate the kth element of x.

When PIVOT = 'F' and DIRECT = 'B' then P is given as the sequence

$$P = P_1, P_2, \ldots, P_n,$$

where P_k is a plane rotation matrix of the $(k, n-1)$ plane designed to annihilate the kth element of x.

When PIVOT = 'V' and DIRECT = 'F' then P is given as the sequence

$$P = P_n, P_{n-1}, \ldots, P_1,$$

where P_k is a plane rotation matrix of the $(k, k+1)$ plane designed to annihilate the kth element of x.

The two by two plane rotation part of P_k has the form given by equation (1) for F06FQF and (10) (with c real) for F06HQF. The cosine and sine that define P_k are returned in C(k) and S(k) respectively and the tangent, $t_k = \mathrm{S}(k)/\mathrm{C}(k)$, is overwritten on the element of X corresponding to x_k.

Note that for routine F06HQF, if the imaginary part of α is supplied as zero, then the imaginary part of α will also be zero on return.

F06HRF generates the parameters θ and z of a complex Householder transformation as described in Section 2.2.4. The elements of z are overwritten on x and β is overwritten on α. Note that $\mathrm{Im}(\beta) = 0$. If x is such that

$$\max_i(|\,\mathrm{Re}(x_i)|, |\,\mathrm{Im}(x_i)|) \leq \max(eps.\max(|\,\mathrm{Re}(\alpha)|, |\,\mathrm{Im}(\alpha)|), tol),$$

where *eps* is the ***machine precision***, then θ is returned such that $\mathrm{Re}(\theta) \leq 0$, as described at the end of Section 2.2.4, otherwise θ is such that

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

with

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

F06FSF generates the parameters ζ and z of a real Householder transformation of the form (14), where μ satisfies (15). The elements of z are overwritten on x and β is overwritten on α. If the elements of x are all zero or are all less than $tol.|\alpha|$, then ζ is returned as zero, otherwise ζ satisfies

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

If *tol* is outside the range $[0, 1]$, then the value zero is used in place of *tol*, but *tol* is unchanged on exit.

F06FRF generates the parameters ζ and z of a real Householder transformation of the form (14), where μ satisfies (16). The elements of z are overwritten on x and β is overwritten on α. If the elements x satisfy

$$\max_i |x_i| \leq \max(eps.|\alpha|, tol),$$

where *eps* is the ***machine precision***, then ζ is returned as zero, otherwise ζ satisfies

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

F06FUF, **F06FTF** and **F06HTF** perform elementary reflections given by

$$\begin{pmatrix} \delta \\ y \end{pmatrix} \leftarrow P \begin{pmatrix} \delta \\ y \end{pmatrix},$$

where P is an elementary reflector. F06FUF is intended for use in conjunction with routine F06FSF, and F06FTF and F06HTF are intended for use in conjunction with routines F06FRF and F06HRF respectively. Note that F06HTF can be used to perform the transformation

$$\begin{pmatrix} \delta \\ y \end{pmatrix} \leftarrow P^H \begin{pmatrix} \delta \\ y \end{pmatrix},$$

by calling F06HTF with CONJG(THETA) in place of THETA.

4.3 The Level-2 Matrix-vector Routines

The matrix-vector routines all have one array argument representing a matrix; usually this is a two-dimensional array but in some cases the matrix is represented by a one-dimensional array.

The size of the matrix is determined by the arguments M and N for an m by n rectangular matrix; and by the argument N for an n by n symmetric, Hermitian, or triangular matrix. Note that it is permissible to call the routines with M or N = 0, in which case the routines exit immediately without referencing their array arguments. For band matrices, the bandwidth is determined by the arguments KL and KU for a rectangular matrix with kl sub-diagonals and ku super-diagonals; and by the argument K for a symmetric, Hermitian, or triangular matrix with k sub-diagonals and/or super-diagonals.

The description of the matrix consists either of the array name (A) followed by the first dimension of the array as declared in the calling (sub)program (LDA), when the matrix is being stored in a two-dimensional array; or the array name (AP) alone when the matrix is being stored as a (packed) vector. In the former case the actual array must contain at least $((n-1)d+l)$ elements, where d is the first dimension of the array, $d \geq l$, and $l = m$ for arrays representing general matrices, $l = n$ for arrays representing symmetric, Hermitian and triangular matrices, $l = kl + ku + 1$ for arrays representing general band matrices and $l = k+1$ for symmetric, Hermitian and triangular band matrices. For one-dimensional arrays representing matrices (**packed storage**) the actual array must contain at least $\frac{1}{2}n(n+1)$ elements.

You may wish to be aware that Chapter F01 provides some utility routines for conversion between storage formats. (See Section 3.3.)

As with the vector routines, vectors are represented by one-dimensional arrays together with a corresponding increment argument (see Section 4.2). The only difference is that for these routines a zero increment is not permitted.

When the vector x consists of k elements then the declared length of the array X in the calling (sub)program must be at least $(1 + (k-1) * |\text{INCX}|)$.

The arguments that specify options are character arguments with the names TRANS, UPLO and DIAG. TRANS is used by the matrix-vector product routines as follows:

Value	**Meaning**
'N'	Operate with the matrix
'T'	Operate with the transpose of the matrix
'C'	Operate with the conjugate transpose of the matrix

In the real case the values 'T', 't', 'C' and 'c' have the same meaning.

UPLO is used by the Hermitian, symmetric, and triangular matrix routines to specify whether the upper or lower triangle is being referenced as follows:

Value	**Meaning**
'U'	Upper triangle
'L'	Lower triangle

DIAG is used by the triangular matrix routines to specify whether or not the matrix is unit triangular, as follows:

Value	**Meaning**
'U'	Unit triangular
'N'	Non-unit triangular

When DIAG is supplied as 'U' the diagonal elements are not referenced.

It is worth noting that actual character arguments in Fortran may be longer than the corresponding dummy arguments. So that, for example, the value 'T' for TRANS may be passed as 'TRANSPOSE'.

The routines for real symmetric and complex Hermitian matrices allow for the matrix to be stored in either the upper (UPLO = 'U') or lower triangle (UPLO = 'L') of a two-dimensional array, or to be packed in a one-dimensional array. In the latter case the upper triangle may be packed sequentially column by column (UPLO = 'U'), or the lower triangle may be packed sequentially column by column

(UPLO = 'L'). Note that for real symmetric matrices packing the upper triangle by column is equivalent to packing the lower triangle by rows, and packing the lower triangle by columns is equivalent to packing the upper triangle by rows. (For complex Hermitian matrices the only difference is that the off-diagonal elements are conjugated.)

For triangular matrices the argument UPLO serves to define whether the matrix is upper (UPLO = 'U') or lower (UPLO = 'L') triangular. In packed storage the triangle has to be packed by column.

The band matrix routines allow storage so that the jth column of the matrix is stored in the jth column of the Fortran array. For a general band matrix the diagonal of the matrix is stored in the $(ku+1)$th row of the array. For a Hermitian or symmetric matrix either the upper triangle (UPLO = 'U') may be stored in which case the leading diagonal is in the $(k+1)$th row of the array, or the lower triangle (UPLO = 'L') may be stored in which case the leading diagonal is in the first row of the array. For an upper triangular band matrix (UPLO = 'U') the leading diagonal is in the $(k+1)$th row of the array and for a lower triangular band matrix (UPLO = 'L') the leading diagonal is in the first row.

For a Hermitian matrix the imaginary parts of the diagonal elements are of course zero and thus the imaginary parts of the corresponding Fortran array elements need not be set, but are assumed to be zero.

For packed triangular matrices the same storage layout is used whether or not DIAG = 'U', i.e., space is left for the diagonal elements even if those array elements are not referenced.

Throughout the following sections A^H denotes the complex conjugate of A^T.

4.3.1 The Level-2 BLAS matrix-vector routines

```
SUBROUTINE        F06PAF     ( TRANS,M,N,      ALPHA,A,LDA,   X,INCX,BETA,Y,INCY )
ENTRY             sgemv      ( TRANS,M,N,      ALPHA,A,LDA,   X,INCX,BETA,Y,INCY )
CHARACTER*1                    TRANS
INTEGER                              M,N,              LDA,     INCX,       INCY
real                                           ALPHA,A(LDA,*),X(*),  BETA,Y(*)

SUBROUTINE        F06SAF     ( TRANS,M,N,      ALPHA,A,LDA,   X,INCX,BETA,Y,INCY )
ENTRY             cgemv      ( TRANS,M,N,      ALPHA,A,LDA,   X,INCX,BETA,Y,INCY )
CHARACTER*1                    TRANS
INTEGER                              M,N,              LDA,     INCX,       INCY
complex                                        ALPHA,A(LDA,*),X(*),  BETA,Y(*)

SUBROUTINE        F06PBF     ( TRANS,M,N,KL,KU,ALPHA,A,LDA,   X,INCX,BETA,Y,INCY )
ENTRY             sgbmv      ( TRANS,M,N,KL,KU,ALPHA,A,LDA,   X,INCX,BETA,Y,INCY )
CHARACTER*1                    TRANS
INTEGER                              M,N,KL,KU,        LDA,     INCX,       INCY
real                                           ALPHA,A(LDA,*),X(*),  BETA,Y(*)

SUBROUTINE        F06SBF     ( TRANS,M,N,KL,KU,ALPHA,A,LDA,   X,INCX,BETA,Y,INCY )
ENTRY             cgbmv      ( TRANS,M,N,KL,KU,ALPHA,A,LDA,   X,INCX,BETA,Y,INCY )
CHARACTER*1                    TRANS
INTEGER                              M,N,KL,KU,        LDA,     INCX,       INCY
complex                                        ALPHA,A(LDA,*),X(*),  BETA,Y(*)

SUBROUTINE        F06PCF     ( UPLO,   N,      ALPHA,A,LDA,   X,INCX,BETA,Y,INCY )
ENTRY             ssymv      ( UPLO,   N,      ALPHA,A,LDA,   X,INCX,BETA,Y,INCY )
CHARACTER*1                    UPLO
INTEGER                                N,              LDA,     INCX,       INCY
real                                           ALPHA,A(LDA,*),X(*),  BETA,Y(*)

SUBROUTINE        F06SCF     ( UPLO,   N,      ALPHA,A,LDA,   X,INCX,BETA,Y,INCY )
ENTRY             chemv      ( UPLO,   N,      ALPHA,A,LDA,   X,INCX,BETA,Y,INCY )
CHARACTER*1                    UPLO
INTEGER                                N,              LDA,     INCX,       INCY
complex                                        ALPHA,A(LDA,*),X(*),  BETA,Y(*)
```

```
SUBROUTINE      F06PDF    ( UPLO,   N,K,    ALPHA,A,LDA,   X,INCX,BETA,Y,INCY )
ENTRY           ssbmv     ( UPLO,   N,K,    ALPHA,A,LDA,   X,INCX,BETA,Y,INCY )
CHARACTER*1                 UPLO
INTEGER                           N,K,            LDA,     INCX,       INCY
real                                        ALPHA,A(LDA,*),X(*),  BETA,Y(*)

SUBROUTINE      F06SDF    ( UPLO,   N,K,    ALPHA,A,LDA,   X,INCX,BETA,Y,INCY )
ENTRY           chbmv     ( UPLO,   N,K,    ALPHA,A,LDA,   X,INCX,BETA,Y,INCY )
CHARACTER*1                 UPLO
INTEGER                           N,K,            LDA,     INCX,       INCY
complex                                     ALPHA,A(LDA,*),X(*),  BETA,Y(*)

SUBROUTINE      F06PEF    ( UPLO,   N,      ALPHA,AP,      X,INCX,BETA,Y,INCY )
ENTRY           sspmv     ( UPLO,   N,      ALPHA,AP,      X,INCX,BETA,Y,INCY )
CHARACTER*1                 UPLO
INTEGER                           N,                       INCX,       INCY
real                                        ALPHA,AP(*),   X(*),  BETA,Y(*)

SUBROUTINE      F06SEF    ( UPLO,   N,      ALPHA,AP,      X,INCX,BETA,Y,INCY )
ENTRY           chpmv     ( UPLO,   N,      ALPHA,AP,      X,INCX,BETA,Y,INCY )
CHARACTER*1                 UPLO
INTEGER                           N,                       INCX,       INCY
complex                                     ALPHA,AP(*),   X(*),  BETA,Y(*)

SUBROUTINE      F06PFF    ( UPLO,TRANS,DIAG,N,    A,LDA,   X,INCX )
ENTRY           strmv     ( UPLO,TRANS,DIAG,N,    A,LDA,   X,INCX )
CHARACTER*1                 UPLO,TRANS,DIAG
INTEGER                                 N,      LDA,     INCX
real                                                A(LDA,*),X(*)

SUBROUTINE      F06SFF    ( UPLO,TRANS,DIAG,N,    A,LDA,   X,INCX )
ENTRY           ctrmv     ( UPLO,TRANS,DIAG,N,    A,LDA,   X,INCX )
CHARACTER*1                 UPLO,TRANS,DIAG
INTEGER                                 N,      LDA,     INCX
complex                                             A(LDA,*),X(*)

SUBROUTINE      F06PGF    ( UPLO,TRANS,DIAG,N,K,  A,LDA,   X,INCX )
ENTRY           stbmv     ( UPLO,TRANS,DIAG,N,K,  A,LDA,   X,INCX )
CHARACTER*1                 UPLO,TRANS,DIAG
INTEGER                                 N,K,    LDA,     INCX
real                                                A(LDA,*),X(*)

SUBROUTINE      F06SGF    ( UPLO,TRANS,DIAG,N,K,  A,LDA,   X,INCX )
ENTRY           ctbmv     ( UPLO,TRANS,DIAG,N,K,  A,LDA,   X,INCX )
CHARACTER*1                 UPLO,TRANS,DIAG
INTEGER                                 N,K,    LDA,     INCX
complex                                             A(LDA,*),X(*)

SUBROUTINE      F06PHF    ( UPLO,TRANS,DIAG,N,    AP,      X,INCX )
ENTRY           stpmv     ( UPLO,TRANS,DIAG,N,    AP,      X,INCX )
CHARACTER*1                 UPLO,TRANS,DIAG
INTEGER                                 N,              INCX
real                                                AP(*),   X(*)

SUBROUTINE      F06SHF    ( UPLO,TRANS,DIAG,N,    AP,      X,INCX )
ENTRY           ctpmv     ( UPLO,TRANS,DIAG,N,    AP,      X,INCX )
CHARACTER*1                 UPLO,TRANS,DIAG
INTEGER                                 N,              INCX
complex                                             AP(*),   X(*)
```

```
SUBROUTINE      F06PJF    ( UPLO,TRANS,DIAG,N,    A,LDA,   X,INCX )
ENTRY           strsv     ( UPLO,TRANS,DIAG,N,    A,LDA,   X,INCX )
CHARACTER*1                 UPLO,TRANS,DIAG
INTEGER                                   N,      LDA,     INCX
real                                            A(LDA,*),X(*)

SUBROUTINE      F06SJF    ( UPLO,TRANS,DIAG,N,    A,LDA,   X,INCX )
ENTRY           ctrsv     ( UPLO,TRANS,DIAG,N,    A,LDA,   X,INCX )
CHARACTER*1                 UPLO,TRANS,DIAG
INTEGER                                   N,      LDA,     INCX
complex                                         A(LDA,*),X(*)

SUBROUTINE      F06PKF    ( UPLO,TRANS,DIAG,N,K,  A,LDA,   X,INCX )
ENTRY           stbsv     ( UPLO,TRANS,DIAG,N,K,  A,LDA,   X,INCX )
CHARACTER*1                 UPLO,TRANS,DIAG
INTEGER                                   N,K,    LDA,     INCX
real                                            A(LDA,*),X(*)

SUBROUTINE      F06SKF    ( UPLO,TRANS,DIAG,N,K,  A,LDA,   X,INCX )
ENTRY           ctbsv     ( UPLO,TRANS,DIAG,N,K,  A,LDA,   X,INCX )
CHARACTER*1                 UPLO,TRANS,DIAG
INTEGER                                   N,K,    LDA,     INCX
complex                                         A(LDA,*),X(*)

SUBROUTINE      F06PLF    ( UPLO,TRANS,DIAG,N,    AP,      X,INCX )
ENTRY           stpsv     ( UPLO,TRANS,DIAG,N,    AP,      X,INCX )
CHARACTER*1                 UPLO,TRANS,DIAG
INTEGER                                   N,               INCX
real                                            AP(*),   X(*)

SUBROUTINE      F06SLF    ( UPLO,TRANS,DIAG,N,    AP,      X,INCX )
ENTRY           ctpsv     ( UPLO,TRANS,DIAG,N,    AP,      X,INCX )
CHARACTER*1                 UPLO,TRANS,DIAG
INTEGER                                   N,               INCX
complex                                         AP(*),   X(*)

SUBROUTINE      F06PMF    (      M,N,ALPHA,               X,INCX,Y,INCY,A,LDA )
ENTRY           sger      (      M,N,ALPHA,               X,INCX,Y,INCY,A,LDA )
INTEGER                          M,N,                       INCX,  INCY,  LDA
real                                 ALPHA,               X(*),  Y(*),  A(LDA,*)

SUBROUTINE      F06SMF    (      M,N,ALPHA,               X,INCX,Y,INCY,A,LDA )
ENTRY           cgeru     (      M,N,ALPHA,               X,INCX,Y,INCY,A,LDA )
INTEGER                          M,N,                       INCX,  INCY,  LDA
complex                              ALPHA,               X(*),  Y(*),  A(LDA,*)

SUBROUTINE      F06SNF    (      M,N,ALPHA,               X,INCX,Y,INCY,A,LDA )
ENTRY           cgerc     (      M,N,ALPHA,               X,INCX,Y,INCY,A,LDA )
INTEGER                          M,N,                       INCX,  INCY,  LDA
complex                              ALPHA,               X(*),  Y(*),  A(LDA,*)

SUBROUTINE      F06PPF    ( UPLO,  N,ALPHA,               X,INCX,       A,LDA )
ENTRY           ssyr      ( UPLO,  N,ALPHA,               X,INCX,       A,LDA )
CHARACTER*1                 UPLO
INTEGER                            N,                       INCX,         LDA
real                                 ALPHA,               X(*),         A(LDA,*)

SUBROUTINE      F06SPF    ( UPLO,  N,ALPHA,               X,INCX,       A,LDA )
ENTRY           cher      ( UPLO,  N,ALPHA,               X,INCX,       A,LDA )
CHARACTER*1                 UPLO
INTEGER                            N,                       INCX,         LDA
real                                 ALPHA
complex                                                     X(*),         A(LDA,*)
```

```
SUBROUTINE      F06PQF    ( UPLO,  N,ALPHA,              X,INCX,       AP )
ENTRY           sspr      ( UPLO,  N,ALPHA,              X,INCX,       AP )
CHARACTER*1                 UPLO
INTEGER                          N,                      INCX
real                                ALPHA,              X(*),         AP(*)

SUBROUTINE      F06SQF    ( UPLO,  N,ALPHA,              X,INCX,       AP )
ENTRY           chpr      ( UPLO,  N,ALPHA,              X,INCX,       AP )
CHARACTER*1                 UPLO
INTEGER                        N,                       INCX
real                              ALPHA
complex                                                 X(*),         AP(*)

SUBROUTINE      F06PRF    ( UPLO,  N,ALPHA,              X,INCX,Y,INCY,A,LDA )
ENTRY           ssyr2     ( UPLO,  N,ALPHA,              X,INCX,Y,INCY,A,LDA )
CHARACTER*1                 UPLO
INTEGER                        N,                       INCX,  INCY,  LDA
real                              ALPHA,              X(*),  Y(*),  A(LDA,*)

SUBROUTINE      F06SRF    ( UPLO,  N,ALPHA,              X,INCX,Y,INCY,A,LDA )
ENTRY           cher2     ( UPLO,  N,ALPHA,              X,INCX,Y,INCY,A,LDA )
CHARACTER*1                 UPLO
INTEGER                        N,                       INCX,  INCY,  LDA
complex                           ALPHA,              X(*),  Y(*),  A(LDA,*)

SUBROUTINE      F06PSF    ( UPLO,  N,ALPHA,              X,INCX,Y,INCY,AP )
ENTRY           sspr2     ( UPLO,  N,ALPHA,              X,INCX,Y,INCY,AP )
CHARACTER*1                 UPLO
INTEGER                        N,                       INCX,  INCY
real                              ALPHA,              X(*),  Y(*),  AP(*)

SUBROUTINE      F06SSF    ( UPLO,  N,ALPHA,              X,INCX,Y,INCY,AP )
ENTRY           chpr2     ( UPLO,  N,ALPHA,              X,INCX,Y,INCY,AP )
CHARACTER*1                 UPLO
INTEGER                        N,                       INCX,  INCY
complex                           ALPHA,              X(*),  Y(*),  AP(*)
```

F06PAF, **F06SAF**, **F06PBF** and **F06SBF** perform the operation

$$\begin{aligned} y &\leftarrow \alpha Ax + \beta y, \quad \text{when TRANS = 'N',} \\ y &\leftarrow \alpha A^T x + \beta y, \quad \text{when TRANS = 'T',} \\ y &\leftarrow \alpha A^H x + \beta y, \quad \text{when TRANS = 'C',} \end{aligned}$$

where A is a general matrix for F06PAF and F06SAF, and is a general band matrix for F06PBF and F06SBF.

F06PCF, **F06SCF**, **F06PEF**, **F06SEF**, **F06PDF** and **F06SDF** perform the operation

$$y \leftarrow \alpha Ax + \beta y$$

where A is symmetric and Hermitian for F06PCF and F06SCF respectively, is symmetric and Hermitian stored in packed form for F06PEF and F06SEF respectively, and is symmetric and Hermitian band for F06PDF and F06SDF.

F06PFF, **F06SFF**, **F06PHF**, **F06SHF**, **F06PGF** and **F06SGF** perform the operation

$$\begin{aligned} x &\leftarrow Ax, \quad \text{when TRANS = 'N',} \\ x &\leftarrow A^T x, \quad \text{when TRANS = 'T',} \\ x &\leftarrow A^H x, \quad \text{when TRANS = 'C',} \end{aligned}$$

where A is a triangular matrix for F06PFF and F06SFF, is a triangular matrix stored in packed form for F06PHF and F06SHF, and is a triangular band matrix for F06PGF and F06SGF.

F06PJF, **F06SJF**, **F06PLF**, **F06SLF**, **F06PKF** and **F06SKF** solve the equations

$$\begin{aligned} Ax &= b, \quad \text{when TRANS = 'N',} \\ A^T x &= b, \quad \text{when TRANS = 'T',} \\ A^H x &= b, \quad \text{when TRANS = 'C',} \end{aligned}$$

where A is a triangular matrix for F06PJF and F06SJF, is a triangular matrix stored in packed form for F06PLF and F06SLF, and is a triangular band matrix for F06PKF and F06SKF. The vector b must be supplied in the array X and is overwritten by the solution. It is important to note that no test for singularity is included in these routines.

F06PMF and **F06SMF** perform the operation

$$A \leftarrow \alpha x y^T + A,$$

where A is a general matrix.

F06SNF performs the operation

$$A \leftarrow \alpha x y^H + A,$$

where A is a general complex matrix.

F06PPF and **F06PQF** perform the operation

$$A \leftarrow \alpha x x^T + A,$$

where A is a symmetric matrix for F06PPF and is a symmetric matrix stored in packed form for F06PQF.

F06SPF and **F06SQF** perform the operation

$$A \leftarrow \alpha x x^H + A,$$

where A is an Hermitian matrix for F06SPF and is an Hermitian matrix stored in packed form for F06SQF.

F06PRF and **F06PSF** perform the operation

$$A \leftarrow \alpha x y^T + \alpha y x^T + A,$$

where A is a symmetric matrix for F06PRF and is a symmetric matrix stored in packed form for F06PSF.

F06SRF and **F06SSF** perform the operation

$$A \leftarrow \alpha x y^H + \bar{\alpha} y x^H + A,$$

where A is an Hermitian matrix for F06SRF and is an Hermitian matrix stored in packed form for F06SSF.

The following argument values are invalid:

Any value of the character arguments DIAG, TRANS, or UPLO whose meaning is not specified.

M < 0

N < 0

KL < 0

KU < 0

K < 0

LDA $<$ M

LDA $<$ KL + KU + 1

LDA $<$ N for the routines involving full Hermitian, symmetric or triangular matrices

LDA $<$ K+1 for the routines involving band Hermitian, symmetric or triangular matrices

INCX $= 0$

INCY $= 0$

If a routine is called with an invalid value 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 is terminated.

4.4 The Level-2 Matrix Routines

The matrix routines have either one or two array arguments representing matrices; currently these are all two-dimensional arrays. When the array name A is used, the conventions for the size and description of the matrix are as for the matrix-vector routines described in Section 4.3. The alternative array name is B which is always followed by its first dimension as declared in the calling (sub)program (LDB).

The array B is used either as the second two-dimensional array argument, or to represent a matrix which is to be multiplied by a sequence of n, $m \times m$ permutation matrices. In this case B represents either an $m \times k$, or a $k \times m$ matrix depending upon whether the permutations are to be applied from the left or the right respectively. n, m and k are represented by the arguments N, M and K. The permutation matrices are represented by a single one-dimensional array argument PERM, which is either an integer or a real array.

Many of the routines in this section are concerned with applying sequences of plane rotations to matrices. For all these routines the sequence of plane rotations is represented by two one-dimensional array arguments, C and S, defining the cosines and sines respectively for each plane rotation (see Section 2.2.1 and Section 2.2.2). In most cases the plane rotations can be restricted to planes $k1$ through to $k2$ ($1 \leq k1 \leq k2 \leq m$ or n) defined by the integer arguments K1 and K2. If any of the above inequalities do not hold, then an immediate return is effected. In the descriptions P^H denotes the complex conjugate of P^T.

Vectors are again represented by one-dimensional array arguments (X or Y) together with a corresponding increment argument (INCX or INCY), which for the routines in this section must be positive.

The character arguments UPLO and TRANS have the same meaning as described in Section 4.3. Additionally five other character arguments are used to specify options in this section, with the names SIDE, PIVOT, DIRECT, NORM and MATRIX. SIDE is used by the permutation routines and many of the plane rotation routines as follows:

Value	**Meaning**
'L'	operate on the left-hand side
'R'	operate on the right-hand side

PIVOT and DIRECT are used together by some of the plane rotation routines. PIVOT is used as follows:

Value	**Meaning**
'B'	bottom (fixed) pivot
'T'	top (fixed) pivot
'V'	variable pivot

and DIRECT is used as follows:

Value	**Meaning**
'B'	backward sequence
'F'	forward sequence

NORM is used by the routines that return the norm of a matrix as follows:

Value	**Meaning**
'M'	$\max_{i,j} \lvert a_{i,j} \rvert$ (element of maximum absolute value. Not strictly a norm)
'1' or 'O'	$\lVert A \rVert_1$ (one norm of a matrix)
'I'	$\lVert A \rVert_\infty$ (infinity norm of a matrix)
'F' or 'E'	$\lVert A \rVert_F$ (Frobenius or Euclidean norm of a matrix)

MATRIX is used by some of the routines to determine the type of matrix represented by the corresponding array argument as follows:

Value	Meaning
'G'	general (rectangular or square) matrix
'U'	upper trapezoidal or triangular matrix
'L'	lower trapezoidal or triangular matrix

For the following routines, WORK must be an array of length M when NORM = 'I', and an array of length 1 otherwise:

F06RAF F06RJF F06UAF F06UJF

For the following routines, WORK must be an array of length N when NORM = 'I', and an array of length 1 otherwise:

F06RBF F06RKF F06RLF F06RMF F06UBF F06UKF
F06ULF F06UMF

For the following routines, WORK must be an array of length N when NORM = 'I', 'l' or 'O', and an array of length 1 otherwise:

F06RCF F06RDF F06REF F06UCF F06UDF F06UEF
F06UFF F06UGF F06UHF

4.4.1 The F06 Level-2 matrix routines

```
SUBROUTINE       F06QFF    ( MATRIX,M,N,           A,LDA,   B,LDB )
CHARACTER*1                  MATRIX
INTEGER                             M,N,             LDA,     LDB
real                                                A(LDA,*),B(LDB,*)

SUBROUTINE       F06TFF    ( MATRIX,M,N,           A,LDA,   B,LDB )
CHARACTER*1                  MATRIX
INTEGER                             M,N,             LDA,     LDB
complex                                             A(LDA,*),B(LDB,*)

SUBROUTINE       F06QHF    ( MATRIX,M,N,CONST,DIAG,A,LDA )
CHARACTER*1                  MATRIX
INTEGER                             M,N,             LDA
real                                    CONST,DIAG,A(LDA,*)

SUBROUTINE       F06THF    ( MATRIX,M,N,CONST,DIAG,A,LDA )
CHARACTER*1                  MATRIX
INTEGER                             M,N,             LDA
complex                                 CONST,DIAG,A(LDA,*)

SUBROUTINE       F06QJF    ( SIDE,TRANS,N,PERM,   K,B,LDB )
CHARACTER*1                  SIDE,TRANS
INTEGER                                 N,PERM(*),K,  LDB
real                                                  B(LDB,*)

SUBROUTINE       F06VJF    ( SIDE,TRANS,N,PERM,   K,B,LDB )
CHARACTER*1                  SIDE,TRANS
INTEGER                                 N,PERM(*),K,  LDB
complex                                               B(LDB,*)

SUBROUTINE       F06QKF    ( SIDE,TRANS,N,PERM,   K,B,LDB )
CHARACTER*1                  SIDE,TRANS
INTEGER                                 N,        K,  LDB
real                                      PERM(*),  B(LDB,*)
```

```
SUBROUTINE            F06VKF    ( SIDE,TRANS,N,PERM,   K,B,LDB )
CHARACTER*1                       SIDE,TRANS
INTEGER                                    N,        K,  LDB
real                                         PERM(*)
complex                                                  B(LDB,*)

SUBROUTINE            F06QMF    ( UPLO,PIVOT,DIRECT,N,K1,K2,C,   S,   A,LDA )
CHARACTER*1                       UPLO,PIVOT,DIRECT
INTEGER                                          N,K1,K2,            LDA
real                                                    C(*),S(*),A(LDA,*)

SUBROUTINE            F06TMF    ( UPLO,PIVOT,DIRECT,N,K1,K2,C,   S,   A,LDA )
CHARACTER*1                       UPLO,PIVOT,DIRECT
INTEGER                                          N,K1,K2,            LDA
real                                                    C(*)
complex                                                      S(*),A(LDA,*)

SUBROUTINE            F06QPF    ( N,ALPHA,X,INCX,Y,INCY,A,LDA,   C,   S )
INTEGER                           N,        INCX,  INCY,  LDA
real                                ALPHA,X(*),  Y(*),  A(LDA,*),C(*),S(*)

SUBROUTINE            F06TPF    ( N,ALPHA,X,INCX,Y,INCY,A,LDA,   C,   S )
INTEGER                           N,        INCX,  INCY,  LDA
complex                             ALPHA,X(*),  Y(*),  A(LDA,*),     S(*)
real                                                          C(*)

SUBROUTINE            F06QQF    ( N,ALPHA,X,INCX,       A,LDA,   C,   S )
INTEGER                           N,        INCX,         LDA
real                                ALPHA,X(*),         A(LDA,*),C(*),S(*)

SUBROUTINE            F06TQF    ( N,ALPHA,X,INCX,       A,LDA,   C,   S )
INTEGER                           N,        INCX,         LDA
complex                             ALPHA,X(*),         A(LDA,*),     S(*)
real                                                              C(*)

SUBROUTINE            F06QRF    ( SIDE,               N,K1,K2,C,   S,   A,LDA )
CHARACTER*1                       SIDE
INTEGER                                               N,K1,K2,            LDA
real                                                          C(*),S(*),A(LDA,*)

SUBROUTINE            F06TRF    ( SIDE,               N,K1,K2,C,   S,   A,LDA )
CHARACTER*1                       SIDE
INTEGER                                               N,K1,K2,            LDA
complex                                                       C(*),     A(LDA,*)
real                                                               S(*)

SUBROUTINE            F06QSF    ( SIDE,               N,K1,K2,C,   S,   A,LDA )
CHARACTER*1                       SIDE
INTEGER                                               N,K1,K2,            LDA
real                                                          C(*),S(*),A(LDA,*)

SUBROUTINE            F06TSF    ( SIDE,               N,K1,K2,C,   S,   A,LDA )
CHARACTER*1                       SIDE
INTEGER                                               N,K1,K2,            LDA
real                                                          C(*)
complex                                                            S(*),A(LDA,*)

SUBROUTINE            F06QTF    ( SIDE,               N,K1,K2,C,   S,   A,LDA )
CHARACTER*1                       SIDE
INTEGER                                               N,K1,K2,            LDA
real                                                          C(*),S(*),A(LDA,*)
```

```
SUBROUTINE            F06TTF    ( SIDE,              N,K1,K2,C,   S,   A,LDA )
CHARACTER*1                       SIDE
INTEGER                                            N,K1,K2,           LDA
real                                                       C(*)
complex                                                         S(*),A(LDA,*)

SUBROUTINE            F06QVF    ( SIDE,              N,K1,K2,C,   S,   A,LDA )
CHARACTER*1                       SIDE
INTEGER                                            N,K1,K2,           LDA
real                                                       C(*),S(*),A(LDA,*)

SUBROUTINE            F06TVF    ( SIDE,              N,K1,K2,C,   S,   A,LDA )
CHARACTER*1                       SIDE
INTEGER                                            N,K1,K2,           LDA
complex                                                    C(*),     A(LDA,*)
real                                                            S(*)

SUBROUTINE            F06QWF    ( SIDE,              N,K1,K2,C,   S,   A,LDA )
CHARACTER*1                       SIDE
INTEGER                                            N,K1,K2,           LDA
real                                                       C(*),S(*),A(LDA,*)

SUBROUTINE            F06TWF    ( SIDE,              N,K1,K2,C,   S,   A,LDA )
CHARACTER*1                       SIDE
INTEGER                                            N,K1,K2,           LDA
real                                                       C(*)
complex                                                         S(*),A(LDA,*)

SUBROUTINE            F06QXF    ( SIDE,PIVOT,DIRECT,M,N,K1,K2,C,   S,   A,LDA )
CHARACTER*1                       SIDE,PIVOT,DIRECT
INTEGER                                       M,N,K1,K2,           LDA
real                                                       C(*),S(*),A(LDA,*)

SUBROUTINE            F06TXF    ( SIDE,PIVOT,DIRECT,M,N,K1,K2,C,   S,   A,LDA )
CHARACTER*1                       SIDE,PIVOT,DIRECT
INTEGER                                       M,N,K1,K2,           LDA
real                                                       C(*)
complex                                                         S(*),A(LDA,*)

SUBROUTINE            F06VXF    ( SIDE,PIVOT,DIRECT,M,N,K1,K2,C,S,      A,LDA )
CHARACTER*1                       SIDE,PIVOT,DIRECT
INTEGER                                       M,N,K1,K2,LDA
real                                                       C(*),S(*)
complex                                                                 A(LDA,*)

SUBROUTINE            F06TYF    ( SIDE,PIVOT,DIRECT,M,N,K1,K2,C,   S,   A,LDA )
CHARACTER*1                       SIDE,PIVOT,DIRECT
INTEGER                                       M,N,K1,K2,           LDA
complex                                                    C(*),     A(LDA,*)
real                                                            S(*)

real FUNCTION         F06RAF    ( NORM,M,N,          A,LDA,     WORK)
CHARACTER*1                       NORM
INTEGER                                M,N,          LDA
real                                                 A(LDA,*),  WORK(*)

real FUNCTION         F06RBF    ( NORM,N,KL,KU,      A,LDA,     WORK)
CHARACTER*1                       NORM
INTEGER                                N,KL,KU,      LDA
real                                                 A(LDA,*),  WORK(*)

real FUNCTION         F06RCF    ( NORM,UPLO,N,       A,LDA,     WORK)
CHARACTER*1                       NORM,UPLO
INTEGER                                     N,       LDA
real                                                 A(LDA,*),  WORK(*)
```

```
real FUNCTION          F06RDF    ( NORM,UPLO,N,       AP,        WORK)
CHARACTER*1                        NORM,UPLO
INTEGER                                  N
real                                                  AP(*),     WORK(*)

real FUNCTION          F06REF    ( NORM,UPLO,N,K,     A,LDA,     WORK)
CHARACTER*1                        NORM,UPLO
INTEGER                                  N,K,       LDA
real                                                  A(LDA,*),  WORK(*)

real FUNCTION          F06RJF    ( NORM,UPLO,DIAG,M,N,A,LDA,     WORK)
CHARACTER*1                        NORM,UPLO,DIAG
INTEGER                                       M,N,  LDA
real                                              A(LDA,*),  WORK(*)

real FUNCTION          F06RKF    ( NORM,UPLO,DIAG,  N,AP,        WORK)
CHARACTER*1                        NORM,UPLO,DIAG
INTEGER                                         N
real                                              AP(*),     WORK(*)

real FUNCTION          F06RLF    ( NORM,UPLO,DIAG,  N,K,A,LDA,   WORK)
CHARACTER*1                        NORM,UPLO,DIAG
INTEGER                                         N,K,  LDA
real                                                  A(LDA,*),WORK(*)

real FUNCTION          F06RMF    ( NORM,            N,  A,LDA,   WORK)
CHARACTER*1                        NORM
INTEGER                                         N,    LDA
real                                                  A(LDA,*),WORK(*)

real FUNCTION          F06UAF    ( NORM,          M,N,  A,LDA,   WORK)
CHARACTER*1                        NORM
INTEGER                                       M,N
real                                                             WORK(*)
complex                                           A(LDA,*)

real FUNCTION          F06UBF    ( NORM,            N,KL,KU,A,LDA,WORK)
CHARACTER*1                        NORM
INTEGER                                         N,KL,KU,  LDA
real                                                             WORK(*)
complex                                                  A(LDA,*)

real FUNCTION          F06UCF    ( NORM,UPLO,       N,      A,LDA,WORK)
CHARACTER*1                        NORM,UPLO
INTEGER                                         N,        LDA
real                                                             WORK(*)
complex                                                  A(LDA,*)

real FUNCTION          F06UDF    ( NORM,UPLO,       N,      AP,   WORK)
CHARACTER*1                        NORM,UPLO
INTEGER                                         N
real                                                             WORK(*)
complex                                                  AP(*)

real FUNCTION          F06UEF    ( NORM,UPLO,       N,K,    A,LDA,WORK)
CHARACTER*1                        NORM,UPLO
INTEGER                                         N,K,      LDA
real                                                             WORK(*)
complex                                                  A(LDA,*)
```

```
real FUNCTION          F06UFF    ( NORM,UPLO,       N,      A,LDA,WORK)
CHARACTER*1                        NORM,UPLO
INTEGER                                             N,        LDA
real                                                              WORK(*)
complex                                                         A(LDA,*)

real FUNCTION          F06UGF    ( NORM,UPLO,       N,      AP,   WORK)
CHARACTER*1                        NORM,UPLO
INTEGER                                             N
real                                                              WORK(*)
complex                                                         AP(*)

real FUNCTION          F06UHF    ( NORM,UPLO,       N,K,    A,LDA,WORK )
CHARACTER*1                        NORM,UPLO
INTEGER                                             N,K,      LDA
real                                                              WORK(*)
complex                                                         A(LDA,*)

real FUNCTION          F06UJF    ( NORM,UPLO,DIAG,M,N,      A,LDA,WORK)
CHARACTER*1                        NORM,UPLO,DIAG
INTEGER                                           M,N,        LDA
real                                                              WORK(*)
complex                                                         A(LDA,*)

real FUNCTION          F06UKF    ( NORM,UPLO,DIAG,  N,      AP,   WORK)
CHARACTER*1                        NORM,UPLO,DIAG
INTEGER                                             N
real                                                              WORK(*)
complex                                                         AP(*)

real FUNCTION          F06ULF    ( NORM,UPLO,DIAG,  N,K,    A,LDA,WORK)
CHARACTER*1                        NORM,UPLO,DIAG
INTEGER                                             N,K,      LDA
real                                                              WORK(*)
complex                                                         A(LDA,*)

real FUNCTION          F06UMF    ( NORM,            N,      A,LDA,WORK)
CHARACTER*1                        NORM
INTEGER                                             N,        LDA
real                                                              WORK(*)
complex                                                         A(LDA,*)
```

F06QFF and **F06TFF** perform the operation

$$B \leftarrow A$$

where A and B are $m \times n$ general (rectangular or square), or upper or lower trapezoidal or triangular matrices.

F06RAF, **F06UAF**, **F06RBF**, **F06UBF**, **F06RCF**, **F06UCF**, **F06RDF**, **F06UDF**, **F06REF**, **F06UEF**, **F06RJF**, **F06UJF**, **F06RKF**, **F06UKF**, **F06RLF**, **F06ULF**, **F06RMF**, **F06UMF**, **F06UFF**, **F06UGF** and **F06UHF** return one of the values $amax$ or $\|A\|_1$ or $\|A\|_\infty$ or $\|A\|_F$ given by

$$\begin{aligned}
amax &= \max_{i,j} |a_{ij}|, \\
\|A\|_1 &= \max_{j} \sum_{i=1}^{m} |a_{ij}|, \\
\|A\|_\infty &= \max_{i} \sum_{j=1}^{n} |a_{ij}|, \\
\|A\|_F &= \left(\sum_{i=1}^{m} \sum_{j=1}^{n} |a_{ij}|^2 \right)^{1/2},
\end{aligned}$$

where A is either an $m \times n$ general rectangular or upper or lower trapezoidal matrix, or a square ($m = n$) band or symmetric or Hermitian or Hessenberg or upper or lower triangular matrix, or a combination of these (e.g. Hermitian band). When A is symmetric or Hermitian or triangular it may be supplied in packed form.

F06QHF and **F06THF** perform the operation

$$a_{ij} \leftarrow \begin{cases} diag, & i = j \\ const, & i \neq j \end{cases}$$

where A is an $m \times n$ general (rectangular or square), or upper or lower trapezoidal or triangular matrix.

F06QJF, **F06VJF**, **F06QKF** and **F06VKF** perform one of the operations

$$\begin{array}{ll} B \leftarrow PB & \text{when SIDE = 'L' and TRANS = 'N',} \\ B \leftarrow P^T B & \text{when SIDE = 'L' and TRANS = 'T',} \\ B \leftarrow BP & \text{when SIDE = 'R' and TRANS = 'N',} \\ B \leftarrow BP^T & \text{when SIDE = 'R' and TRANS = 'T',} \end{array}$$

where P is an $m \times m$ permutation matrix of the form

$$P = P_{1,i_1} P_{2,i_2} \dots P_{n,i_n},$$

P_{j,i_j} being the permutation matrix that interchanges items j and i_j. That is, P_{j,i_j} is the unit matrix with rows and columns j and i_j interchanged. If $j = i_j$ then $P = I$. i_j must satisfy $1 \leq i_j \leq m$. P_{j,i_j} is represented by the jth element of the argument PERM such that PERM(j) = i_j.

When SIDE = 'L', B is an $m \times k$ matrix and when SIDE = 'R', B is a $k \times m$ matrix. Note that m is not actually an argument of these routines.

F06QMF and **F06TMF** perform the operations

$A \leftarrow PAP^T$ for F06QMF and

$A \leftarrow PAP^H$ for F06TMF,

where A is an $n \times n$ symmetric (Hermitian for F06TMF) matrix and P is an orthogonal (unitary for F06TMF) matrix consisting of a sequence of plane rotations, applied in planes $k1$ to $k2$. When DIRECT = 'F' then P is given by the sequence

$$P = P_{k2-1}, \dots, P_{k1+1}, P_{k1}$$

and when DIRECT = 'B' then P is given by the sequence

$$P = P_{k1}, P_{k1+1}, \dots, P_{k2-1},$$

where P_k is a plane rotation matrix for the $(k, k+1)$ plane when PIVOT = 'V', P_k is a plane rotation matrix for the $(k1, k+1)$ plane when PIVOT = 'T' and P_k is a plane rotation matrix for the $(k, k2)$ plane when PIVOT = 'B' or 'b'.

The two by two part of the plane rotations are assumed to be of the form given by equation (1) for F06QMF and (10) (with c real) for F06TMF. The cosine and sine that define P_k must be supplied in C(k) and S(k) respectively.

F06QPF and **F06TPF** perform the factorization

$$\alpha x y^T + U = QR,$$

where α is a scalar, x and y are n element vectors, U and R are $n\times$ upper triangular matrices and Q is an $n \times n$ orthogonal (unitary for F06TPF) matrix. For F06TPF, U must have real diagonal elements and R is returned with real diagonal elements. Q is formed as two sequences of plane rotations, P and S. P is a sequence of the form

$$P = P_1, P_2, \dots, P_{n-1},$$

where P_k is a rotation for the (k, n) plane, chosen so that

$$Px = \beta e_n,$$

e_n being the last column of the unit matrix. S is a sequence of the form

$$S = S_{n-1}, S_{n-2}, \ldots, S_1,$$

where S_k is a rotation for the (k, n) plane, chosen so that

$$S(\alpha\beta e_n y^T + PU) = R$$

Q is given as

$Q^T = SP$ for F06QPF and as
$Q^H = DSP$ for F06TPF,

where D is a unitary diagonal matrix with a non-unit element only in d_n, which is chosen to make r_{nn} real and is returned in S(n).

The two by two part of the plane rotations are of the form of equation (1) for F06QPF and (10) (with c real) for F06TPF. The cosine and sine that define S_k are returned in C(k) and S(k) respectively and the tangent that defines the rotation P_k is returned in the element of X corresponding to x_k. (Routines F06BCF and F06CCF may be used to recover the cosine and sine from a given tangent.) The array Y is unchanged on exit.

U must be supplied in the $n \times n$ upper triangular part of the array A and is overwritten by R. In the case of F06TPF the imaginary parts of the diagonal elements of U must be supplied as zero.

F06QQF and **F06TQF** perform the factorization

$$\begin{pmatrix} U \\ \alpha x^T \end{pmatrix} = Q \begin{pmatrix} R \\ 0 \end{pmatrix},$$

where *alpha* is a scalar, x is an n element vector, U and R are $n \times n$ upper triangular matrices and Q is an $(n+1) \times (n+1)$ orthogonal (unitary for F06TQF) matrix. For F06TQF, if U is supplied with real diagonal elements then the diagonal elements of R will also be real. Q is formed as a sequence of plane rotations

$Q^T = Q_n, \ldots, Q_2, Q_1$ for F06QQF
$Q^H = Q_n, \ldots, Q_2, Q_1$ for F06TQF,

where Q_k is a rotation for the $(k, n+1)$ plane.

The two by two part of the plane rotations are of the form of equation (1) for F06QQF and (10) (with c real) for F06TQF. The cosine, sine and tangent that define Q_k are returned respectively in C(k), S(k) and the element of X that corresponds to x_k. (Routines F06BCF and F06CCF may be used to recover the cosine and sine from a given tangent.)

U must be supplied in the $n \times n$ upper triangular part of the array A and is overwritten by R.

F06QRF and **F06TRF** perform one of the operations

$$\begin{array}{ll} R \leftarrow PH & \text{when SIDE = 'L',} \\ \left.\begin{array}{l} R \leftarrow HP^T \text{ for F06QRF} \\ R \leftarrow HP^H \text{ for F06TRF} \end{array}\right\} & \text{when SIDE = 'R',} \end{array}$$

where H is an $n \times n$ upper Hessenberg matrix, P is an $n \times n$ orthogonal (unitary for F06TRF) matrix and R is an $n \times n$ upper triangular matrix. H is assumed to have (possibly) non-zero sub-diagonal elements only in elements $h_{k+1,k}$ and these elements must be supplied in S(k), $k = k1, k1+1, \ldots, k2+1$. For F06TRF, H must have real sub-diagonal elements and R will be returned with real diagonal elements. The upper triangular part of H must be supplied in A and is overwritten by R.

When SIDE = 'L', P is given by

$P = P_{k2-1}, P_{k2-2}, \ldots, P_{k1}$ for F06QRF,
$P = D_{k2}, P_{k2-1}, P_{k2-2}, \ldots, P_{k1}$ for F06TRF,

and when SIDE = 'R', P is given by

$P = P_{k1}, P_{k1+1}, \ldots, P_{k2-1}$ for F06QRF,
$P = D_{k1}, P_{k1}, P_{k1+1}, \ldots, P_{k2-1}$ for F06TRF,

where P_k is a plane rotation matrix for the $(k, k+1)$ plane and D_k is a unitary diagonal matrix with a non-unit diagonal element only in d_k.

The two by two part of the plane rotations are of the form of equation (1) for F06QRF and (11) (with s real) for F06TRF. The cosine and sine that define P_k are returned in C(k) and S(k) respectively and, for F06TRF, d_k is returned in C($k2$).

F06QSF and **F06TSF** perform one of the operations

$$\left.\begin{array}{l} R \leftarrow PH \\ R \leftarrow HP^T \text{ for F06QSF} \\ R \leftarrow HP^H \text{ for F06TSF} \end{array}\right\} \begin{array}{l} \text{when SIDE = 'L',} \\ \text{when SIDE = 'R',} \end{array}$$

where H is an $n \times n$ upper spiked matrix, P is an $n \times n$ orthogonal (unitary for F06TSF) matrix and R is an $n \times n$ upper triangular matrix. When SIDE = 'L', H is assumed to have a row spike and have (possibly) non-zero sub-diagonal elements only in elements $h_{k2,k}$, $k = k1, k1+1, \ldots, k2-1$ and when SIDE = 'R', H is assumed to have a column spike and have (possibly) non-zero sub-diagonal elements only in elements $h_{k+1,k1}$, $k = k1, k1+1, \ldots, k2-1$. These spiked elements must be supplied in the elements S(k), $k = k1, k1+1, \ldots, k2-1$. For F06TSF, H must have real diagonal elements except in the position where the spike joins the diagonal, that is $h_{k2,k2}$ for a row spike and $h_{k1,k1}$ for a column spike, and R will be returned with real diagonal elements. The upper triangular part of H must be supplied in A and is overwritten by R.

When SIDE = 'L', P is given by

$P = P_{k2-1}, P_{k2-2}, \ldots, P_{k1}$ for F06QSF,
$P = D_{k2}, P_{k2-1}, P_{k2-2}, \ldots, P_{k1}$ for F06TSF,

where P_k is a plane rotation matrix for the $(k, k2)$ plane and when SIDE = 'R', P is given by

$P = P_{k1+1}, P_{k1+2}, \ldots, P_{k2-1}$ for F06QSF,
$P = D_{k1}, P_{k1+1}, P_{k1+2}, \ldots, P_{k2+1}$ for F06TSF,

where P_k is a plane rotation matrix for the $(k1, k)$ plane and D_k is a unitary diagonal matrix with a non-unit diagonal element only in d_k.

The two by two part of the plane rotations are of the form of equation (1) for F06QSF and (10) (with c real) for F06TSF. The cosine and sine that define P_k are returned in C(k) and S(k) respectively and, for F06TSF, d_k is returned in S($k2$).

F06QTF and **F06TTF** perform one of the operations

$$\left.\begin{array}{l} R \leftarrow PUQ^T \text{ for F06QTF} \\ R \leftarrow PUQ^H \text{ for F06TTF} \end{array}\right\} \text{ when SIDE = 'L'}$$

$$\left.\begin{array}{l} R \leftarrow QUP^T \text{ for F06QTF} \\ R \leftarrow QUP^H \text{ for F06TTF} \end{array}\right\} \text{ when SIDE = 'R'}$$

where U and R are $n \times n$ upper triangular matrices and P and Q are $n \times n$ orthogonal (unitary for F06TTF) matrices, with P given. When SIDE = 'L' then P is assumed to be given by

$$P = P_{k2-1}, P_{k2-2}, \ldots, P_{k1}$$

and Q is then given as

$$Q = Q_{k2-1}, Q_{k2-2}, \ldots, Q_{k1}$$

and when SIDE = 'R' then P is assumed to be given by

$$P = P_{k1}, P_{k1+1}, \ldots, P_{k2-1}$$

and Q is then given as

$$Q = Q_{k1}, Q_{k1+1}, \ldots, Q_{k2-1},$$

where P_k and Q_k are plane rotations matrices for the $(k, k+1)$ plane.

The two by two part of the plane rotations are of the form of equation (1) for F06QTF and (10) (with c real) for F06TTF. The cosine and sine that define P_k must be supplied in C(k) and S(k) respectively, and are overwritten by the cosine and sine that define Q_k.

The matrix U must be supplied in the upper triangular part of A and is overwritten by R.

F06QVF and **F06TVF** perform one of the operations

$$\left.\begin{array}{l} H \leftarrow PU \\ H \leftarrow UP^T \text{ for F06QVF} \\ H \leftarrow UP^H \text{ for F06TVF} \end{array}\right\} \begin{array}{l} \text{when SIDE = 'L',} \\ \text{when SIDE = 'R',} \end{array}$$

where U is an $n \times n$ upper triangular matrix, H is an $n \times n$ upper Hessenberg matrix and P is an $n \times n$ orthogonal (unitary for F06TVF) matrix. For F06TVF, U must be supplied with real diagonal elements and H will have real sub-diagonal elements. When SIDE = 'L' or 'l', then P is assumed to be given by

$$P = P_{k1}, P_{k1+1}, \ldots, P_{k2-1}$$

and when SIDE = 'R', then P is assumed to be given by

$$P = P_{k2-1}, P_{k2-2}, \ldots, P_{k1},$$

where P_k is a plane rotation matrix for the $(k, k+1)$ plane.

The two by two part of the plane rotations are of the form of equation (1) for F06QVF and (11) (with s real) for F06TVF. The cosine and sine that define P_k must be supplied in C(k) and S(k) respectively.

The matrix U must be supplied in the upper triangular part of A and is overwritten by the upper triangular part of H. The sub-diagonal elements of H, $h_{k+1,k}$, are returned in the elements S(k), $k = k1, k1+1, \ldots, k2-1$.

F06QWF and **F06TWF** perform one of the operations

$$\left.\begin{array}{l} H \leftarrow PU \\ H \leftarrow UP^T \text{ for F06QWF} \\ H \leftarrow UP^H \text{ for F06TWF} \end{array}\right\} \begin{array}{l} \text{when SIDE = 'L',} \\ \text{when SIDE = 'R',} \end{array}$$

where U is an $n \times n$ upper triangular matrix, H is an $n \times n$ upper spiked matrix and P is an $n \times n$ orthogonal (unitary for F06TWF) matrix. For F06TWF, U must be supplied with real diagonal elements and H will have real diagonal elements, except for the position where the spike joins the diagonal.

When SIDE = 'L', then P is assumed to be given by

$$P = P_{k1}, P_{k1+1}, \ldots, P_{k2-1},$$

where P_k is a plane rotation matrix for the $(k, k2)$ plane and when SIDE = 'R', then P is assumed to be given by

$$P = P_{k2-1}, P_{k2-2}, \ldots, P_{k1},$$

where P_k is a plane rotation matrix for the $(k1, k+1)$ plane.

The two by two part of the plane rotations are of the form of equation (1) for F06QWF and (10) for F06TWF. The cosine and sine that define P_k must be supplied in the elements C(k) and S(k) respectively.

The matrix U must be supplied in the upper triangular part of A and is overwritten by the upper triangular part of H. The sub-diagonal elements of H, $h_{k2,k}$ (row spike) when SIDE = 'L', and $h_{k+1,k1}$ (column spike) when SIDE = 'R' are returned in the elements S(k), $k = k1, k1+1, \ldots, k2-1$.

F06QXF, **F06TXF**, **F06TYF** and **F06VXF** perform the operation

$$\left.\begin{array}{l} A \leftarrow PA \\ A \leftarrow AP^T \text{ for F06QXF and F06VXF} \\ A \leftarrow AP^H \text{ for F06TXF and F06TYF} \end{array}\right\} \begin{array}{l} \text{when SIDE = 'L',} \\ \text{when SIDE = 'R',} \end{array}$$

where A is an m by n matrix and P is an orthogonal (unitary for F06TXF and F06TYF) matrix consisting of a sequence of plane rotations, applied in planes $k1$ to $k2$. When DIRECT = 'F' then P is given by the sequence

$$P = P_{k2-1}, \ldots, P_{k1+1}, P_{k1}$$

and when DIRECT = 'B' then P is given by the sequence

$$P = P_{k1}, P_{k1+1}, \ldots, P_{k2-1},$$

where P_k is a plane rotation matrix for the $(k, k+1)$ plane when PIVOT = 'V', P_k is a plane rotation matrix for the $(k1, k+1)$ plane, when PIVOT = 'T' and P_k is a plane rotation matrix for the $(k, k2)$ plane when PIVOT = 'B' or 'b'.

The two by two plane rotation part of P_k is assumed to be of the form given by equation (1) for F06QXF and F06VXF, (10) (with c real) for F06TXF and (11) (with s real) for F06TYF. The cosine and sine that define P_k must be supplied in C(k) and S(k) respectively.

4.5 The Level-3 Matrix-matrix Routines

The matrix-matrix routines all have either two or three arguments representing a matrix, one of which is an input-output argument, and in each case the arguments are two-dimensional arrays.

The sizes of the matrices are determined by one or more of the arguments M, N and K. The size of the input-output array is always determined by the arguments M and N for a rectangular m by n matrix, and by the argument N for a square n by n matrix. It is permissible to call the routines with M or N = 0, in which case the routines exit immediately without referencing their array arguments.

Many of the routines perform an operation of the form

$$C \leftarrow P + \beta C,$$

where P is the product of two matrices, or the sum of two such products. When the inner dimension of the matrix product is different from m or n it is denoted by K. Again it is permissible to call the routines with K = 0 and if M > 0, but K = 0, then the routines perform the operation

$$C \leftarrow \beta C.$$

As with the Level-2 routines (see Section 4) the description of the matrix consists of the array name (A or B or C) followed by the first dimension (LDA or LDB or LDC).

The arguments that specify options are character arguments with the names SIDE, TRANSA, TRANSB, TRANS, UPLO and DIAG. UPLO and DIAG have the same values and meanings as for the Level-2 routines (see Section 4.3); TRANSA, TRANSB and TRANS have the same values and meanings as TRANS in the Level-2 routines, where TRANSA and TRANSB apply to the matrices A and B respectively. SIDE is used by the routines as follows:

Value	Meaning
'L'	Multiply general matrix by symmetric, Hermitian or triangular matrix on the left
'R'	Multiply general matrix by symmetric, Hermitian or triangular matrix on the right

The storage conventions for matrices are as for the Level-2 routines (see Section 4.3).

4.5.1 The Level-3 BLAS matrix-matrix routines

```
SUBROUTINE      F06YAF     ( TRANSA,TRANSB,       M,N,K,ALPHA,A,LDA,   B,LDB,   BETA,C,LDC )
ENTRY           sgemm      ( TRANSA,TRANSB,       M,N,K,ALPHA,A,LDA,   B,LDB,   BETA,C,LDC )
CHARACTER*1                  TRANSA,TRANSB
INTEGER                                           M,N,K,        LDA,     LDB,          LDC
real                                                    ALPHA,A(LDA,*),B(LDB,*),BETA,C(LDC,*)

SUBROUTINE      F06ZAF     ( TRANSA,TRANSB,       M,N,K,ALPHA,A,LDA,   B,LDB,   BETA,C,LDC )
ENTRY           cgemm      ( TRANSA,TRANSB,       M,N,K,ALPHA,A,LDA,   B,LDB,   BETA,C,LDC )
CHARACTER*1                  TRANSA,TRANSB
INTEGER                                           M,N,K,        LDA,     LDB,          LDC
complex                                                 ALPHA,A(LDA,*),B(LDB,*),BETA,C(LDC,*)
```

```
SUBROUTINE          F06YCF    ( SIDE,UPLO,            M,N,  ALPHA,A,LDA,   B,LDB,   BETA,C,LDC )
ENTRY               ssymm     ( SIDE,UPLO,            M,N,  ALPHA,A,LDA,   B,LDB,   BETA,C,LDC )
CHARACTER*1                     SIDE,UPLO
INTEGER                                                 M,N,          LDA,     LDB,          LDC
real                                                          ALPHA,A(LDA,*),B(LDB,*),BETA,C(LDC,*)

SUBROUTINE          F06ZCF    ( SIDE,UPLO,            M,N,  ALPHA,A,LDA,   B,LDB,   BETA,C,LDC )
ENTRY               chemm     ( SIDE,UPLO,            M,N,  ALPHA,A,LDA,   B,LDB,   BETA,C,LDC )
CHARACTER*1                     SIDE,UPLO
INTEGER                                                 M,N,          LDA,     LDB,          LDC
complex                                                       ALPHA,A(LDA,*),B(LDB,*),BETA,C(LDC,*)

SUBROUTINE          F06ZTF    ( SIDE,UPLO,            M,N,  ALPHA,A,LDA,   B,LDB,   BETA,C,LDC )
ENTRY               csymm     ( SIDE,UPLO,            M,N,  ALPHA,A,LDA,   B,LDB,   BETA,C,LDC )
CHARACTER*1                     SIDE,UPLO
INTEGER                                                 M,N,          LDA,     LDB,          LDC
complex                                                       ALPHA,A(LDA,*),B(LDB,*),BETA,C(LDC,*)

SUBROUTINE          F06YFF    ( SIDE,UPLO,TRANSA,DIAG,M,N,  ALPHA,A,LDA,   B,LDB )
ENTRY               strmm     ( SIDE,UPLO,TRANSA,DIAG,M,N,  ALPHA,A,LDA,   B,LDB )
CHARACTER*1                     SIDE,UPLO,TRANSA,DIAG
INTEGER                                                 M,N,          LDA,     LDB
real                                                          ALPHA,A(LDA,*),B(LDB,*)

SUBROUTINE          F06ZFF    ( SIDE,UPLO,TRANSA,DIAG,M,N,  ALPHA,A,LDA,   B,LDB )
ENTRY               ctrmm     ( SIDE,UPLO,TRANSA,DIAG,M,N,  ALPHA,A,LDA,   B,LDB )
CHARACTER*1                     SIDE,UPLO,TRANSA,DIAG
INTEGER                                                 M,N,          LDA,     LDB
complex                                                       ALPHA,A(LDA,*),B(LDB,*)

SUBROUTINE          F06YJF    ( SIDE,UPLO,TRANSA,DIAG,M,N,  ALPHA,A,LDA,   B,LDB )
ENTRY               strsm     ( SIDE,UPLO,TRANSA,DIAG,M,N,  ALPHA,A,LDA,   B,LDB )
CHARACTER*1                     SIDE,UPLO,TRANSA,DIAG
INTEGER                                                 M,N,          LDA,     LDB
real                                                          ALPHA,A(LDA,*),B(LDB,*)

SUBROUTINE          F06ZJF    ( SIDE,UPLO,TRANSA,DIAG,M,N,  ALPHA,A,LDA,   B,LDB )
ENTRY               ctrsm     ( SIDE,UPLO,TRANSA,DIAG,M,N,  ALPHA,A,LDA,   B,LDB )
CHARACTER*1                     SIDE,UPLO,TRANSA,DIAG
INTEGER                                                 M,N,          LDA,     LDB
complex                                                       ALPHA,A(LDA,*),B(LDB,*)

SUBROUTINE          F06YPF    ( UPLO,TRANS,             N,K,ALPHA,A,LDA,            BETA,C,LDC )
ENTRY               ssyrk     ( UPLO,TRANS,             N,K,ALPHA,A,LDA,            BETA,C,LDC )
CHARACTER*1                     UPLO,TRANS
INTEGER                                                 N,K,          LDA,                  LDC
real                                                          ALPHA,A(LDA,*),           BETA,C(LDC,*)

SUBROUTINE          F06ZPF    ( UPLO,TRANS,             N,K,ALPHA,A,LDA,            BETA,C,LDC )
ENTRY               cherk     ( UPLO,TRANS,             N,K,ALPHA,A,LDA,            BETA,C,LDC )
CHARACTER*1                     UPLO,TRANS
INTEGER                                                 N,K,          LDA,                  LDC
real                                                          ALPHA,                    BETA
complex                                                             A(LDA,*),                 C(LDC,*)

SUBROUTINE          F06ZUF    ( UPLO,TRANS,             N,K,ALPHA,A,LDA,            BETA,C,LDC )
ENTRY               csyrk     ( UPLO,TRANS,             N,K,ALPHA,A,LDA,            BETA,C,LDC )
CHARACTER*1                     UPLO,TRAN
INTEGER                                                 N,K,          LDA,                  LDC
complex                                                       ALPHA,A(LDA,*),           BETA,C(LDC,*)

SUBROUTINE          F06YRF    ( UPLO,TRANS,             N,K,ALPHA,A,LDA,   B,LDB,   BETA,C,LDC )
ENTRY               ssyr2k    ( UPLO,TRANS,             N,K,ALPHA,A,LDA,   B,LDB,   BETA,C,LDC )
CHARACTER*1                     UPLO,TRANS
INTEGER                                                 N,K,          LDA,     LDB,          LDC
real                                                          ALPHA,A(LDA,*),B(LDB,*),BETA,C(LDC,*)
```

```
SUBROUTINE          F06ZRF    ( UPLO,TRANS,        N,K,ALPHA,A,LDA,   B,LDB,   BETA,C,LDC )
ENTRY               cher2k    ( UPLO,TRANS,        N,K,ALPHA,A,LDA,   B,LDB,   BETA,C,LDC )
CHARACTER*1                     UPLO,TRANS
INTEGER                                          N,K,        LDA,     LDB,          LDC
real                                                                             BETA
complex                                              ALPHA,A(LDA,*),B(LDB,*),     C(LDC,*)

SUBROUTINE          F06ZWF    ( UPLO,TRANS,        N,K,ALPHA,A,LDA,   B,LDB,   BETA,C,LDC )
ENTRY               csyr2k    ( UPLO,TRANS,        N,K,ALPHA,A,LDA,   B,LDB,   BETA,C,LDC )
CHARACTER*1                     UPLO,TRANS
INTEGER                                          N,K,        LDA,     LDB,          LDC
complex                                              ALPHA,A(LDA,*),B(LDB,*),BETA,C(LDC,*)
```

F06YAF and **F06ZAF** perform the operation indicated in the following table:

	TRANSA = 'N'	TRANSA = 'T'	TRANSA = 'C'
TRANSB = 'N'	$C \leftarrow \alpha AB + \beta C$	$C \leftarrow \alpha A^T B + \beta C$	$C \leftarrow \alpha A^H B + \beta C$
	A is $m \times k$, B is $k \times n$	A is $k \times m$, B is $k \times n$	A is $k \times m$, B is $k \times n$
TRANSB = 'T'	$C \leftarrow \alpha AB^T + \beta C$	$C \leftarrow \alpha A^T B^T + \beta C$	$C \leftarrow \alpha A^H B^T + \beta C$
	A is $m \times k$, B is $n \times k$	A is $k \times m$, B is $n \times k$	A is $k \times m$, B is $n \times k$
TRANSB = 'C'	$C \leftarrow \alpha AB^H + \beta C$	$C \leftarrow \alpha A^T B^H + \beta C$	$C \leftarrow \alpha A^H B^H + \beta C$
	A is $m \times k$, B is $n \times k$	A is $k \times m$, B is $n \times k$	A is $k \times m$, B is $n \times k$

where A and B are general matrices and C is a general m by n matrix.

F06YCF, **F06ZCF** and **F06ZTF** perform the operation indicated in the following table:

SIDE = 'L'	SIDE = 'R'
$C \leftarrow \alpha AB + \beta C$	$C \leftarrow \alpha BA + \beta C$
A is $m \times m$	B is $m \times n$
B is $m \times n$	A is $n \times n$

where A is symmetric for F06YCF and F06ZTF and is Hermitian for F06ZCF, B is a general matrix and C is a general m by n matrix.

F06YFF and **F06ZFF** perform the operation indicated in the following table:

	TRANSA = 'N'	TRANSA = 'T'	TRANSA = 'C'
SIDE = 'L'	$B \leftarrow \alpha AB$	$B \leftarrow \alpha A^T B$	$B \leftarrow \alpha A^H B$
	A is triangular $m \times m$	A is triangular $m \times m$	A is triangular $m \times m$
SIDE = 'R'	$B \leftarrow \alpha BA$	$B \leftarrow \alpha BA^T$	$B \leftarrow \alpha BA^H$
	A is triangular $n \times n$	A is triangular $n \times n$	A is triangular $n \times n$

where B is a general m by n matrix.

F06YJF and **F06ZJF** solve the equations, indicated in the following table, for X:

	TRANSA = 'N'	TRANSA = 'T'	TRANSA = 'C'
SIDE = 'L'	$AX = \alpha B$	$A^T X = \alpha B$	$A^H X = \alpha B$
	A is triangular $m \times m$	A is triangular $m \times m$	A is triangular $m \times m$
SIDE = 'R'	$XA = \alpha B$	$XA^T = \alpha B$	$XA^H = \alpha B$
	A is triangular $n \times n$	A is triangular $n \times n$	A is triangular $n \times n$

where B is a general m by n matrix. The m by n solution matrix X is overwritten on the array B. It is important to note that no test for singularity is included in these routines.

F06YPF, **F06ZPF** and **F06ZUF** perform the operation indicated in the following table:

	TRANS = 'N'	TRANS = 'T'	TRANS = 'C'
F06YPF	$C \leftarrow \alpha AA^T + \beta C$	$C \leftarrow \alpha A^T A + \beta C$	$C \leftarrow \alpha A^T A + \beta C$
F06ZUF	$C \leftarrow \alpha AA^T + \beta C$	$C \leftarrow \alpha A^T A + \beta C$	–
F06ZPF	$C \leftarrow \alpha AA^H + \beta C$	–	$C \leftarrow \alpha A^H A + \beta C$
	A is $n \times k$	A is $k \times n$	A is $k \times n$

where A is a general matrix and C is an n by n symmetric matrix for F06YPF and F06ZUF, and is an n by n Hermitian matrix for F06ZPF.

F06YRF, **F06ZRF** and **F06ZWF** perform the operation indicated in the following table:

	TRANS = 'N'	TRANS = 'T'	TRANS = 'C'
F06YRF	$C \leftarrow \alpha AB^T + \alpha BA^T + \beta C$	$C \leftarrow \alpha A^T B + \alpha B^T A + \beta C$	$C \leftarrow \alpha A^T B + \alpha B^T A + \beta C$
F06ZWF	$C \leftarrow \alpha AB^T + \alpha BA^T + \beta C$	$C \leftarrow \alpha A^T B + \alpha B^T A + \beta C$	–
F06ZRF	$C \leftarrow \alpha AB^H + \bar{\alpha} BA^H + \beta C$	–	$C \leftarrow \alpha A^H B + \bar{\alpha} B^H A + \beta C$
	A and B are $n \times k$	A and B are $k \times n$	A and B are $k \times n$

where A and B are general matrices and C is an n by n symmetric matrix for F06YRF and F06ZWF, and is an n by n Hermitian matrix for F06ZPF.

The following values of arguments are invalid:

Any value of the character arguments SIDE, TRANSA, TRANSB, TRANS, UPLO or DIAG, whose meaning is not specified.

M < 0

N < 0

K < 0

LDA $<$ the number of rows in the matrix A

LDB $<$ the number of rows in the matrix B

LDC $<$ the number of rows in the matrix C

If a routine is called with an invalid value 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 is terminated.

5 Routines Withdrawn or Scheduled for Withdrawal

Since Mark 13 the following routines have been withdrawn. Advice on replacing calls to these routines is given in the document 'Advice on Replacement Calls for Withdrawn/Superseded Routines'.

F06QGF F06VGF

6 Indexes of BLAS routines

Real Matrices		
BLAS single precision	**BLAS double precision**	**NAG**
ISAMAX	IDAMAX	F06JLF
SASUM	DASUM	F06EKF
SAXPY	DAXPY	F06ECF
SAXPYI	DAXPYI	F06ETF
SCASUM	DCASUM	F06JKF
SCNRM2	DCNRM2	F06JJF
SCOPY	DCOPY	F06EFF
SDOT	DDOT	F06EAF
SDOTI	DDOTI	F06ERF
SGBMV	DGBMV	F06PBF
SGEMM	DGEMM	F06YAF
SGEMV	DGEMV	F06PAF
SGER	DGER	F06PMF
SGTHR	DGTHR	F06EUF
SGTHRZ	DGTHRZ	F06EVF
SNRM2	DNRM2	F06EJF
SROT	DROT	F06EPF
SROTG	DROTG	F06AAF
SROTI	DROTI	F06EXF
SSBMV	DSBMV	F06PDF
SSCAL	DSCAL	F06EDF
SSCTR	DSCTR	F06EWF
SSPMV	DSPMV	F06PEF
SSPR	DSPR	F06PQF
SSPR2	DSPR2	F06PSF
SSWAP	DSWAP	F06EGF
SSYMM	DSYMM	F06YCF
SSYMV	DSYMV	F06PCF
SSYR	DSYR	F06PPF
SSYR2	DSYR2	F06PRF
SSYR2K	DSYR2K	F06YRF
SSYRK	DSYRK	F06YPF
STBMV	DTBMV	F06PGF
STBSV	DTBSV	F06PKF
STPMV	DTPMV	F06PHF
STPSV	DTPSV	F06PLF
STRMM	DTRMM	F06YFF
STRMV	DTRMV	F06PFF
STRSM	DTRSM	F06YJF
STRSV	DTRSV	F06PJF

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

7 References

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

[2] 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

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

[4] 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

[5] 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

[6] Golub G H and Van Loan C F (1989) *Matrix Computations* Johns Hopkins University Press (2nd Edition), Baltimore

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