NAG Library Chapter Introduction

F16 – Further 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. Most routines for such operations conform either to the specications of the BLAS (Basic Linear Algebra Subprograms) or to the specifications of the BLAST (Basic Linear Algebra Subprograms Technical) Forum. This chapter includes routines from the BLAST specifications. Most (BLAS) routines for such operations are available in Chapter F06.

2   Background to the Problems

Most of the routines in this chapter meet the specification of Basic Linear Algebra Subprograms Technical (BLAST) Forum (2001).

They are called extensively by routines in other chapters of the NAG Library, especially in the linear algebra chapters. They are intended to be useful building-blocks for users of the Library who are developing their own applications. The routines fall into four main groups (following the definitions introduced by the BLAS):

Level 0: scalar operations;

Level 1: vector operations;

Level 2: matrix-vector operations and matrix operations which includes single matrix operations;

Level 3: matrix-matrix operations.

The terminology reflects the number of operations involved, so for example a Level 2 routine involves $O(n^2)$ operations, for vectors and matrices of order n.

Because of the overlap of functionality with Chapter F06, only a subset of BLAST routines are implemented in this chapter. A full descripion of the

3   Recommendations on Choice and Use of Available Routines

3.1   Naming Scheme

3.1.1   NAG names

Table 1 shows the naming scheme for the routines in this chapter which follows the naming scheme used in Chapter F06.

	Level-0	Level-1	Level-2	Level-3
integer	–	F16D_F	–	–
'real'	–	F16E_F	–	–
'real'	–	–	F16R_F	–
'complex'	–	F16G_F	–	–
'complex'	–	–	F16U_F	–
'mixed type'	–	F16J_F	–	–

Table 1

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

3.1.2   BLAS names

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

In many implementations of the NAG Library, references to Chapter F06 BLAS names may be linked to an efficient machine-specific implementation of the BLAS, usually provided by the vendor of the machine; Chapter F16 BLAS routines are unlikely to be provide by a vendor. Such implementations are

stringently tested before being used with the NAG Library, to ensure that they correctly meet the specifications of the BLAS, and that they return the desired accuracy. Use of BLAS names is recommended for efficiency.

References to NAG routine names (beginning F06- or F16-) are always linked to the code provided in the NAG Library and may be significantly slower (in the case of Chapter F06 routines) than the equivalent BLAS routine.

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

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

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

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

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

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

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

    GE   general

    GB   general band

    SY   symmetric

    SP   symmetric (packed storage)

    SB   symmetric band

    HE   (complex) Hermitian

    HP   (complex) Hermitian (packed storage)

    HB   (complex) Hermitian band

    TR   triangular

    TP   triangular (packed storage)

    TB   triangular band

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

    MV   matrix-vector product

    MM   matrix-matrix product

    R   rank-1 update

    R2   rank-2 update

    RK   rank-k update

    R2K rank-$2k$ update

    SV   solve a system of linear equations

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

Thus the routine BLAS_DAXPBY performs a sum of two real, scaled vectors in double precision; the corresponding routine for complex scalars and vectors is BLAS_ZAXPBY.

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

3.2 The Level-0 Scalar Routines

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

3.3   The Level-1 Vector Routines

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

3.4   The Level-2 Matrix-vector and Matrix Routines

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

3.5   The Level-3 Matrix-matrix Routines

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

3.6   Vector Arguments

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

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

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

Negative increments are permitted only for:

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

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

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

3.7   Matrix Arguments and Storage Schemes

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

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

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

> – **band storage** for band matrices;

> – storage for spiked matrices.

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

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

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

3.7.1 Conventional storage

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

3.7.2 Packed storage

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

3.7.3 Rectangular Full Packed (RFP) storage

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

3.7.4 Band storage

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

3.7.5 Unit triangular matrices

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

3.7.6 Real diagonal elements of complex Hermitian matrices

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

3.8 Option Parameters

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

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

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

The following option parameters are used in this chapter:

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

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

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

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

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

If DIAG = 'U', unit triangular;

if DIAG = 'N', nonunit triangular.

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

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

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

if NORM = 'I', ∞-norm of a matrix;

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

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

3.8.1 Matrix norms

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

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

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

Infinity-norm (NORM = 'I'):

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

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

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

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

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

3.9   Error Handling

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

If one of the Level-2 or Level-3 BLAS routines is called with an invalid value of one of its parameters, 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 parameter, and execution of the program is terminated. The following values of parameters are invalid:

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

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

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

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

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

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

4   Functionality Index

5   Auxiliary Routines Associated with Library Routine Parameters

None.

6   Routines Withdrawn or Scheduled for Withdrawal

None.

7   References

Basic Linear Algebra Subprograms Technical (BLAST) Forum (2001) *Basic Linear Algebra Subprograms Technical (BLAST) Forum Standard* University of Tennessee, Knoxville, Tennessee http://www.netlib.org/blas/blast-forum/blas-report.pdf

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

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

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

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

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

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

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