NAG Library Chapter Introduction

f16 – NAG Interface to BLAS

Contents

1 Scope of the Chapter

This chapter is concerned with basic linear algebra functions which perform elementary algebraic operations involving scalars, vectors and matrices. Most functions for such operations conform either to the specifications of the BLAS (Basic Linear Algebra Subprograms) or to the specifications of the BLAST (Basic Linear Algebra Subprograms Technical) Forum. This chapter includes functions conforming to both specifications. Two additional functions for such operations are available in Chapter f06.

2 Background to the Problems

Most of the functions in this chapter meet the specification of the BLAS as described in Basic Linear Algebra Subprograms Technical (BLAST) Forum (2001), Lawson *et al.* (1979), Dongarra *et al.* (1988) and Dongarra *et al.* (1990).

They are called extensively by functions in other chapters of the NAG C 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 functions fall into four main groups (following the definitions introduced by the BLAS):

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 function involves $O(n^2)$ operations, for vectors and matrices of order n.

In many implementations of the NAG C Library, the BLAS functions in this chapter serve as interfaces to an efficient machine-specific implementation of the BLAS, usually provided by the vendor of the machine. Such implementations are stringently tested before being used with the NAG C Library, to ensure that they correctly meet the specifications of the BLAS, and that they return the desired accuracy.

2.1 The Use of BLAS Names

Many of the functions in other chapters of the Library call the functions in this chapter, and in particular a number of the BLAS are called. These functions 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 functions. If you are in any doubt about how to avoid this, please consult the NAG Technical Support Service.

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

3 Recommendations on Choice and Use of Available Functions

3.1 Naming Scheme

3.1.1 NAG names

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

		Level-1	Level-2	Level-3
integer	Chapter f16 function	f16d_c	–	–
'real'	BLAS function	–	f16p_c	f16y_c
'real'	Chapter f16 function	f16f_c	f16q_c f16r_c	–
'real'	BLAST function	f16e_c	f16q_c	–

			f16r_c	
‘complex’	BLAS function	–	f16s_c	f16z_c
‘complex’	Chapter f16 function	f16h_c	f16t_c f16u_c	–
‘complex’	BLAST function	f16g_c	f16t_c f16u_c	–
‘mixed type’	BLAS function	f16j_c	–	–

Table 1

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

3.2 The Level-1 Vector Functions

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

3.3 The Level-2 Matrix-vector and Matrix Functions

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

3.4 The Level-3 Matrix-matrix Functions

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

3.5 Vector Arguments

Vector arguments are represented by a one-dimensional array, immediately followed by an **increment** argument whose name consists of the three characters INC followed by the name of the array. For example, a vector x is represented by the two arguments **x** and **incx**. The length of the vector, n say, is passed as a separate argument, **n**.

The increment argument is the spacing (stride) in the array between the elements of the vector. For instance, if **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 $\mathbf{x}(1 + (i-1) \times \mathbf{incx})$. When **incx** ≤ 0, the elements are stored in the reverse order so that the vector element x_i is in the array element $\mathbf{x}(1 - (n-i) \times \mathbf{incx})$ and hence, in particular, the element x_n is in $\mathbf{x}(1)$. The declared length of the array **x** in the calling function must be at least $(1 + (\mathbf{n} - 1) \times |\mathbf{incx}|)$.

Negative increments are permitted only for:

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

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

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

3.6 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;

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 functions in Chapters f01, f02, f03 and f04.)

Chapter f01 provides some utility functions for conversion between storage schemes.

3.6.1 Conventional storage

Please see Section 3.3.1 in the f07 Chapter Introduction for full details.

3.6.2 Packed storage

Please see Section 3.3.2 in the f07 Chapter Introduction for full details.

3.6.3 Rectangular Full Packed (RFP) storage

Please see Section 3.3.3 in the f07 Chapter Introduction for full details.

3.6.4 Band storage

Please see Section 3.3.4 in the f07 Chapter Introduction for full details.

3.6.5 Unit triangular matrices

Please see Section 3.3.5 in the f07 Chapter Introduction for full details.

3.6.6 Real diagonal elements of complex Hermitian matrices

Please see Section 3.3.6 in the f07 Chapter Introduction for full details.

3.7 Option Arguments

In addition to the order argument of type **Nag_OrderType**, most functions in this Chapter have one or more option arguments of various types; only options of the correct type may be supplied.

The following option arguments are used in this chapter:

If **trans** = NoTranspose, operate with the matrix (Not transposed);

if **trans** = Transpose, operate with the Transpose of the matrix;

if **trans** = ConjugateTranspose, operate with the Conjugate transpose of the matrix.

If **uplo** = Nag_Upper, upper triangle or trapezoid of matrix;

if **uplo** = Nag_Lower, lower triangle or trapezoid of matrix.

If **diag** = Nag_UnitDiag, unit triangular;

if **diag** = NotUnitTriangular, nonunit triangular.

If **side** = LeftSide, operate from the left-hand side;

if **side** = RightSide, operate from the right-hand side.

If **norm** = Nag_OneNorm, 1-norm of a matrix;

if **norm** = Nag_InfNorm, ∞-norm of a matrix;

if **norm** = Nag_FrobeniusNorm, Frobenius or Euclidean norm of a matrix;

if **norm** = Nag_MaxNorm, maximum absolute value of the elements of a matrix (not strictly a norm).

3.7.1 Matrix norms

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

One-norm (**norm** = Nag_OneNorm):

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

Infinity-norm (**norm** = Nag_InfNorm):

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

Frobenius or Euclidean norm (**norm** = Nag_FrobeniusNorm):

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

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

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

3.8 Error Handling

Functions in this chapter use the usual NAG C Library error-handling.

4 Functionality Index

5 Auxiliary Functions Associated with Library Function Arguments

None.

6 Functions 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

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