A Proposed Specification of BLAS Routines in C

S.P. Datardina, J.J. Du Croz, S.J. Hammarling and M.W. Pont

NAG Limited
Wilkinson House
Jordan Hill Road
OXFORD
United Kingdom, OX2 8DR
NAG Technical Report TR6/90 - NP2125

Abstract

This report proposes specifications for a set of C equivalents to the Fortran specifications for the Basic Linear Algebra Subprograms (BLAS).

1 Introduction

In 1973, Hanson, Krogh and Lawson described the advantages of adopting a set of basic routines for problems in linear algebra [6]. The original basic linear algebra subprograms, commonly referred to as the BLAS or, in view of subsequent developments, the Level 1 BLAS, have been very successful and have been used in a wide range of software including LINPACK [3]. An excellent discussion of the *raison d'être* of the BLAS is given in Dodson and Lewis, [2].

Subsequently a set of Level 2 BLAS for matrix-vector operations, motivated by the development of vector-processing machines, was proposed by Dongarra *et al.*, [4]. More recently a set of Level 3 BLAS for matrix-matrix operations, motivated by the development of hierarchical memory and parallel machines, has been proposed by Dongarra *et al.*, [5].

All three levels of BLAS are now included in the NAG Fortran Library [8].

NAG has recently launched a C Library [7], and in the continuing development of that library the question of C equivalents of the Fortran BLAS naturally arises. This report proposes a specification for a set of basic linear algebra functions in C. We would welcome comments on the proposal.

2 BLAS in C

One immediate question arises as to the acronym for the basic linear algebra functions in C. The logical choice would perhaps be the BLAFs, but since BLAS has come to be thought of as a generic name, we propose that they be referred to as the BLAS in C, or the C BLAS.

The proposed argument lists follow closely the equivalent Fortran ones, the main changes being that enumeration types are used instead of character types for option specification, and leading dimension parameters (e.g. `LDA`) are replaced by trailing dimension parameters (e.g. `tda`). Two dimensional arrays are therefore stored in an analogous fashion to those in Fortran rather than using the alternative approach of vectors of pointers to vectors, which would force the user to perform rather a large amount of work when initialising them.

It has been suggested that the names of the C versions of the BLAS should be different from the Fortran versions, because sometimes both Fortran and C versions will be available, and some systems do not distinguish between Fortran and C names. The distinction could be made, for example, simply by prefixing each name with 'c_'. This would lead to names of up to eight characters, though only the first six are guaranteed unique by the draft ANSI C standard. In this document, the usual Fortran names have been used.

The argument lists use the following data types:

- `Integer`: an integer data type of at least 32 bits.
- `float`: the regular single precision floating-point type.
- `double`: the regular double precision floating-point type.
- `Scomplex`: a single precision complex type.
- `Complex`: a double precision complex type.

plus the enumeration types given by

```
typedef enum { NoTranspose, Transpose, ConjugateTranspose } MatrixTranspose;
typedef enum { UpperTriangle, LowerTriangle } MatrixTriangle;
typedef enum { UnitTriangular, NotUnitTriangular } MatrixUnitTriangular;
typedef enum { LeftSide, RightSide } OperationSide;
```

The format of the names of the enum types, using phrases with capitalised initial letters and no underscores, is intended to follow one common system employed in C software.

It is envisaged that the complex data types would be stored in Cartesian form, i.e. as real and imaginary parts; for example

```
typedef struct {
   double re, im;
} Complex;
```

The operations performed by the C BLAS are intended to be identical to those performed by the corresponding Fortran BLAS, as specified in [2], [4] and [5].

We have not proposed C equivalents of two of the Level 1 BLAS – DQDOTI and DQDOTA – which require quadruple precision arithmetic.

Since the publication of [2], [4] and [5], there have been some suggestions for adding to the Fortran BLAS – for example, Level 1 routines for complex plane rotations, or Level 2 routines for complex symmetric matrices. We have not proposed adding such routines to the C BLAS at this stage, because we hope that a 'standard' set of additions to the Fortran BLAS may be specified as a by-product of the LAPACK project [1]. It would then be straightforward to propose corresponding C BLAS, along the same lines as the routines proposed here.

References

[1] Anderson, E., Bai, Z., Bischof, C., Demmel, J., Dongarra, J.J., Du Croz, J., Greenbaum, A., Hammarling, S., McKenney, A. and Sorensen, D. (1990). *LAPACK: A Portable Linear Algebra Library for High-Performance Computers* LAPACK Working Note No. 20, University of Tennessee, Technical Report CS-90-105, Department of Computer Science, 107 Ayres Hall, Knoxville, TN 37996-1301.

[2] Dodson, D. and Lewis, J. (1985). *Issues Relating to Extension of the Basic Linear Algebra Subprograms* ACM Signum Newsletter, **20**, pp. 2–18.

[3] Dongarra, J.J., Bunch, J.R., Moler, C.B. and Stewart, G.W. (1978). LINPACK Users' Guide. SIAM, Philadelphia.

[4] Dongarra, J.J., Du Croz, J., Hammarling, S. and Hanson, R.J. (1988). *An Extended Set of FORTRAN Basic Linear Algebra Subprograms* ACM Trans. Math. Softw., **14**, pp. 1–32.

[5] Dongarra, J.J., Du Croz, J., Duff, I.S. and Hammarling, S. (1990). *A Set of Level 3 Basic Linear Algebra Subprograms* ACM Trans. Math. Softw., **16**, pp. 1–28.

[6] Hanson, R., Krogh, F. and Lawson, C. (1973). *A Proposal for Standard Linear Algebra Subprograms* ACM Signum Newsletter, **8**, p. 16.

[7] The NAG C Library, Mark 1. (1990). The Numerical Algorithms Group Ltd, Oxford.

[8] The NAG Fortran Library, Mark 15. (1991). The Numerical Algorithms Group Ltd, Oxford.

Level 1 BLAS in C

generate a plane rotation

```
void  srotg ( float *a,  float *b,  float *c,  float *s)
void  drotg (double *a, double *b, double *c, double *s)
```

generate a modified plane rotation

```
void srotmg ( float *d1,  float *d2,  float *a,  float b,
              float *param)
void drotmg (double *d1, double *d2, double *a, double b,
             double *param)
```

apply a plane rotation

```
void   srot (Integer n,   float *x, Integer incx,   float *y,
             Integer incy,   float c,   float s)
void   drot (Integer n,  double *x, Integer incx,  double *y,
             Integer incy,  double c,  double s)
```

apply a modified plane rotation

```
void  srotm (Integer n,   float *x, Integer incx,   float *y,
             Integer incy,   float *param)
void  drotm (Integer n,  double *x, Integer incx,  double *y,
             Integer incy,  double *param)
```

$x \leftrightarrow y$

```
void  sswap (Integer n,    float *x, Integer incx,
                float *y, Integer incy)
void  dswap (Integer n,   double *x, Integer incx,
               double *y, Integer incy)
void  cswap (Integer n, Scomplex *x, Integer incx,
             Scomplex *y, Integer incy)
void  zswap (Integer n,  Complex *x, Integer incx,
              Complex *y, Integer incy)
```

$x \leftarrow \alpha x$

```
void  sscal (Integer n,    float alpha,    float *x, Integer incx)
void  dscal (Integer n,   double alpha,   double *x, Integer incx)
void  cscal (Integer n, Scomplex alpha, Scomplex *x, Integer incx)
void  zscal (Integer n,  Complex alpha,  Complex *x, Integer incx)
void csscal (Integer n,    float alpha, Scomplex *x, Integer incx)
void zdscal (Integer n,   double alpha,  Complex *x, Integer incx)
```

$y \leftarrow x$

```
    void  scopy (Integer n,    float *x, Integer incx,
                    float *y, Integer incy)
    void  dcopy (Integer n,   double *x, Integer incx,
                   double *y, Integer incy)
    void  ccopy (Integer n, Scomplex *x, Integer incx,
                 Scomplex *y, Integer incy)
    void  zcopy (Integer n,  Complex *x, Integer incx,
                  Complex *y, Integer incy)
```

$y \leftarrow \alpha x + y$

```
    void  saxpy (Integer n,    float alpha,    float *x,
                 Integer incx,    float *y, Integer incy)
    void  daxpy (Integer n,   double alpha,   double *x,
                 Integer incx,   double *y, Integer incy)
    void  caxpy (Integer n, Scomplex alpha, Scomplex *x,
                 Integer incx, Scomplex *y, Integer incy)
    void  zaxpy (Integer n,  Complex alpha,  Complex *x,
                 Integer incx,  Complex *y, Integer incy)
```

$dot \leftarrow x^T y$

```
   float   sdot (Integer n,    float *x, Integer incx,
                    float *y, Integer incy)
  double  dsdot (Integer n,    float *x, Integer incx,
                    float *y, Integer incy)
  double   ddot (Integer n,   double *x, Integer incx,
                   double *y, Integer incy)
```

$dot \leftarrow \alpha + x^T y$

```
   float sdsdot (Integer n, float alpha, float *x, Integer incx,
                 float *y, Integer incy)
```

$dotu \leftarrow x^T y$

```
Scomplex  cdotu (Integer n, Scomplex *x, Integer incx,
                 Scomplex *y, Integer incy)
 Complex  zdotu (Integer n,  Complex *x, Integer incx,
                  Complex *y, Integer incy)
```

$dotc \leftarrow x^H y$

```
Scomplex  cdotc (Integer n, Scomplex *x, Integer incx,
                 Scomplex *y, Integer incy)
 Complex  zdotc (Integer n,  Complex *x, Integer incx,
                  Complex *y, Integer incy)
```

$nrm2 \leftarrow \|x\|_2$

```
  float  snrm2 (Integer n,    float *x, Integer incx)
 double  dnrm2 (Integer n,   double *x, Integer incx)
  float scnrm2 (Integer n, Scomplex *x, Integer incx)
 double dznrm2 (Integer n,  Complex *x, Integer incx)
```

$asum \leftarrow \|x\|_1$

```
  float  sasum (Integer n,    float *x, Integer incx)
 double  dasum (Integer n,   double *x, Integer incx)
```

$asum \leftarrow \|\mathrm{Re}\,(x)\|_1 + \|\mathrm{Im}\,(x)\|_1$

```
  float scasum (Integer n, Scomplex *x, Integer incx)
 double dzasum (Integer n,  Complex *x, Integer incx)
```

$amax \leftarrow$ first k such that $|x_k| = \max|x_i|$

```
Integer isamax (Integer n,    float *x, Integer incx)
Integer idamax (Integer n,   double *x, Integer incx)
```

$amax \leftarrow$ first k such that $|\mathrm{Re}(x_k)| + |\mathrm{Im}(x_k)| = \max(|\mathrm{Re}(x_i)| + |\mathrm{Im}(x_i)|)$

```
Integer icamax (Integer n, Scomplex *x, Integer incx)
Integer izamax (Integer n,  Complex *x, Integer incx)
```

Level 2 BLAS in C

$y \leftarrow \alpha Ax + \beta y$, $y \leftarrow \alpha A^T x + \beta y$, $y \leftarrow \alpha A^H x + \beta y$, A is m by n

```
void  sgemv (MatrixTranspose trans, Integer m, Integer n,
                float alpha,    float *a, Integer tda,
                float *x, Integer incx,    float beta,
                float *y, Integer incy)
void  dgemv (MatrixTranspose trans, Integer m, Integer n,
               double alpha,   double *a, Integer tda,
               double *x, Integer incx,   double beta,
               double *y, Integer incy)
void  cgemv (MatrixTranspose trans, Integer m, Integer n,
             Scomplex alpha, Scomplex *a, Integer tda,
             Scomplex *x, Integer incx, Scomplex beta,
             Scomplex *y, Integer incy)
void  zgemv (MatrixTranspose trans, Integer m, Integer n,
              Complex alpha,  Complex *a, Integer tda,
              Complex *x, Integer incx,  Complex beta,
              Complex *y, Integer incy)
```

$y \leftarrow \alpha Ax + \beta y$, $y \leftarrow \alpha A^T x + \beta y$, $y \leftarrow \alpha A^H x + \beta y$, A is m by n band

```
void  sgbmv (MatrixTranspose trans, Integer m, Integer n,
             Integer kl, Integer ku,    float alpha,
                float *a, Integer tda,    float x, Integer incx,
                float beta,    float *y, Integer incy)
void  dgbmv (MatrixTranspose trans, Integer m, Integer n,
             Integer kl, Integer ku,   double alpha,
               double *a, Integer tda,   double x, Integer incx,
               double beta,   double *y, Integer incy)
void  cgbmv (MatrixTranspose trans, Integer m, Integer n,
             Integer kl, Integer ku, Scomplex alpha,
             Scomplex *a, Integer tda, Scomplex x, Integer incx,
             Scomplex beta, Scomplex *y, Integer incy)
void  zgbmv (MatrixTranspose trans, Integer m, Integer n,
             Integer kl, Integer ku,  Complex alpha,
              Complex *a, Integer tda,  Complex x, Integer incx,
              Complex beta,  Complex *y, Integer incy)
```

$y \leftarrow \alpha Ax + \beta y$, A is n by n Hermitian

```
void  chemv (MatrixTriangle uplo, Integer n, Scomplex alpha,
             Scomplex *a, Integer tda, Scomplex *x, Integer incx,
             Scomplex beta, Scomplex *y, Integer incy)
void  zhemv (MatrixTriangle uplo, Integer n,  Complex alpha,
              Complex *a, Integer tda,  Complex *x, Integer incx,
              Complex beta,  Complex *y, Integer incy)
```

$y \leftarrow \alpha Ax + \beta y$, A is n by n Hermitian band

```
void  chbmv (MatrixTriangle uplo, Integer n, Integer k, Scomplex alpha,
             Scomplex *a, Integer tda, Scomplex *x, Integer incx,
             Scomplex beta, Scomplex *y, Integer incy)
void  zhbmv (MatrixTriangle uplo, Integer n, Integer k,  Complex alpha,
              Complex *a, Integer tda,  Complex *x, Integer incx,
              Complex beta,  Complex *y, Integer incy)
```

$y \leftarrow \alpha Ax + \beta y$, A is n by n Hermitian in packed form

```
void  chpmv (MatrixTriangle uplo, Integer n, Scomplex alpha,
             Scomplex *ap, Scomplex *x, Integer incx,
             Scomplex beta, Scomplex *y, Integer incy)
void  zhpmv (MatrixTriangle uplo, Integer n,  Complex alpha,
              Complex *ap,  Complex *x, Integer incx,
              Complex beta,  Complex *y, Integer incy)
```

$y \leftarrow \alpha Ax + \beta y$, A is n by n symmetric

```
void  ssymv (MatrixTriangle uplo, Integer n,    float alpha,
                float *a, Integer tda,    float *x, Integer incx,
                float beta,    float *y, Integer incy)
void  dsymv (MatrixTriangle uplo, Integer n,   double alpha,
               double *a, Integer tda,   double *x, Integer incx,
               double beta,   double *y, Integer incy)
```

$y \leftarrow \alpha Ax + \beta y$, A is n by n symmetric band

```
void  ssbmv (MatrixTriangle uplo, Integer n, Integer k,    float alpha,
                float *a, Integer tda,    float *x, Integer incx,
                float beta,    float *y, Integer incy)
void  dsbmv (MatrixTriangle uplo, Integer n, Integer k,   double alpha,
               double *a, Integer tda,   double *x, Integer incx,
               double beta,   double *y, Integer incy)
```

$y \leftarrow \alpha Ax + \beta y$, A is n by n symmetric in packed form

```
void  sspmv (MatrixTriangle uplo, Integer n,    float alpha,
                float *ap,    float *x, Integer incx,
                float beta,    float *y, Integer incy)
void  dspmv (MatrixTriangle uplo, Integer n,   double alpha,
               double *ap,   double *x, Integer incx,
               double beta,   double *y, Integer incy)
```

$x \leftarrow Ax$, $x \leftarrow A^T x$, $x \leftarrow A^H x$, A is n by n triangular

```
void  strmv (MatrixTriangle uplo, MatrixTranspose trans,
             MatrixUnitTriangular diag, Integer n,    float *a,
             Integer tda,    float *x, Integer incx)
```

```
void  dtrmv (MatrixTriangle uplo, MatrixTranspose trans,
             MatrixUnitTriangular diag, Integer n,   double *a,
             Integer tda,   double *x, Integer incx)
void  ctrmv (MatrixTriangle uplo, MatrixTranspose trans,
             MatrixUnitTriangular diag, Integer n, Scomplex *a,
             Integer tda, Scomplex *x, Integer incx)
void  ztrmv (MatrixTriangle uplo, MatrixTranspose trans,
             MatrixUnitTriangular diag, Integer n,  Complex *a,
             Integer tda,  Complex *x, Integer incx)
```

$x \leftarrow Ax$, $x \leftarrow A^T x$, $x \leftarrow A^H x$, A is n by n triangular band

```
void  stbmv (MatrixTriangle uplo, MatrixTranspose trans,
             MatrixUnitTriangular diag, Integer n, Integer k,
                float *a, Integer tda,    float *x, Integer incx)
void  dtbmv (MatrixTriangle uplo, MatrixTranspose trans,
             MatrixUnitTriangular diag, Integer n, Integer k,
               double *a, Integer tda,   double *x, Integer incx)
void  ctbmv (MatrixTriangle uplo, MatrixTranspose trans,
             MatrixUnitTriangular diag, Integer n, Integer k,
             Scomplex *a, Integer tda, Scomplex *x, Integer incx)
void  ztbmv (MatrixTriangle uplo, MatrixTranspose trans,
             MatrixUnitTriangular diag, Integer n, Integer k,
              Complex *a, Integer tda,  Complex *x, Integer incx)
```

$x \leftarrow Ax$, $x \leftarrow A^T x$, $x \leftarrow A^H x$, A is n by n triangular in packed form

```
void  stpmv (MatrixTriangle uplo, MatrixTranspose trans,
             MatrixUnitTriangular diag, Integer n,
                float *ap,    float *x, Integer incx)
void  dtpmv (MatrixTriangle uplo, MatrixTranspose trans,
             MatrixUnitTriangular diag, Integer n,
               double *ap,   double *x, Integer incx)
void  ctpmv (MatrixTriangle uplo, MatrixTranspose trans,
             MatrixUnitTriangular diag, Integer n,
             Scomplex *ap, Scomplex *x, Integer incx)
void  ztpmv (MatrixTriangle uplo, MatrixTranspose trans,
             MatrixUnitTriangular diag, Integer n,
              Complex *ap,  Complex *x, Integer incx)
```

$x \leftarrow A^{-1}x$, $x \leftarrow A^{-T}x$, $x \leftarrow A^{-H}x$, A is n by n triangular

```
void  strsv (MatrixTriangle uplo, MatrixTranspose trans,
             MatrixUnitTriangular diag, Integer n,
                float *a, Integer tda,    float *x, Integer incx)
void  dtrsv (MatrixTriangle uplo, MatrixTranspose trans,
             MatrixUnitTriangular diag, Integer n,
               double *a, Integer tda,   double *x, Integer incx)
```

```
void  ctrsv (MatrixTriangle uplo, MatrixTranspose trans,
             MatrixUnitTriangular diag, Integer n,
             Scomplex *a, Integer tda, Scomplex *x, Integer incx)
void  ztrsv (MatrixTriangle uplo, MatrixTranspose trans,
             MatrixUnitTriangular diag, Integer n,
              Complex *a, Integer tda,  Complex *x, Integer incx)
```

$x \leftarrow A^{-1}x$, $x \leftarrow A^{-T}x$, $x \leftarrow A^{-H}x$, A is n by n triangular band

```
void  stbsv (MatrixTriangle uplo, MatrixTranspose trans,
             MatrixUnitTriangular diag, Integer n, Integer k,
                float *a, Integer tda,    float *x, Integer incx)
void  dtbsv (MatrixTriangle uplo, MatrixTranspose trans,
             MatrixUnitTriangular diag, Integer n, Integer k,
               double *a, Integer tda,   double *x, Integer incx)
void  ctbsv (MatrixTriangle uplo, MatrixTranspose trans,
             MatrixUnitTriangular diag, Integer n, Integer k,
             Scomplex *a, Integer tda, Scomplex *x, Integer incx)
void  ztbsv (MatrixTriangle uplo, MatrixTranspose trans,
             MatrixUnitTriangular diag, Integer n, Integer k,
              Complex *a, Integer tda,  Complex *x, Integer incx)
```

$x \leftarrow A^{-1}x$, $x \leftarrow A^{-T}x$, $x \leftarrow A^{-H}x$, A is n by n triangular in packed form

```
void  stpsv (MatrixTriangle uplo, MatrixTranspose trans,
             MatrixUnitTriangular diag, Integer n,
                float *ap,    float *x, Integer incx)
void  dtpsv (MatrixTriangle uplo, MatrixTranspose trans,
             MatrixUnitTriangular diag, Integer n,
               double *ap,   double *x, Integer incx)
void  ctpsv (MatrixTriangle uplo, MatrixTranspose trans,
             MatrixUnitTriangular diag, Integer n,
             Scomplex *ap, Scomplex *x, Integer incx)
void  ztpsv (MatrixTriangle uplo, MatrixTranspose trans,
             MatrixUnitTriangular diag, Integer n,
              Complex *ap,  Complex *x, Integer incx)
```

$A \leftarrow \alpha xy^T + A$, A is m by n

```
void   sger (Integer m, Integer n,    float alpha,
                float *x, Integer incx,    float *y, Integer incy,
                float *a, Integer tda)
void   dger (Integer m, Integer n,   double alpha,
               double *x, Integer incx,   double *y, Integer incy,
               double *a, Integer tda)
```

$A \leftarrow \alpha x y^T + A$, A is m by n

```
void  cgeru (Integer m, Integer n, Scomplex alpha,
             Scomplex *x, Integer incx, Scomplex *y, Integer incy,
             Scomplex *a, Integer tda)
void  zgeru (Integer m, Integer n,  Complex alpha,
              Complex *x, Integer incx,  Complex *y, Integer incy,
              Complex *a, Integer tda)
```

$A \leftarrow \alpha x y^H + A$, A is m by n

```
void  cgerc (Integer m, Integer n, Scomplex alpha,
             Scomplex *x, Integer incx, Scomplex *y, Integer incy,
             Scomplex *a, Integer tda)
void  zgerc (Integer m, Integer n,  Complex alpha,
              Complex *x, Integer incx,  Complex *y, Integer incy,
              Complex *a, Integer tda)
```

$A \leftarrow \alpha x x^H + A$, A is n by n Hermitian

```
void   cher (MatrixTriangle uplo, Integer n, Scomplex alpha,
             Scomplex *x, Integer incx, Scomplex *a, Integer tda)
void   zher (MatrixTriangle uplo, Integer n,  Complex alpha,
              Complex *x, Integer incx,  Complex *a, Integer tda)
```

$A \leftarrow \alpha x x^H + A$, A is n by n Hermitian in packed form

```
void   chpr (MatrixTriangle uplo, Integer n, Scomplex alpha,
             Scomplex *x, Integer incx, Scomplex *ap)
void   zhpr (MatrixTriangle uplo, Integer n,  Complex alpha,
              Complex *x, Integer incx,  Complex *ap)
```

$A \leftarrow \alpha x y^H + y(\alpha x)^H + A$, A is n by n Hermitian

```
void  cher2 (MatrixTriangle uplo, Integer n, Scomplex alpha,
             Scomplex *x, Integer incx, Scomplex *y, Integer incy,
             Scomplex *a, Integer tda)
void  zher2 (MatrixTriangle uplo, Integer n,  Complex alpha,
              Complex *x, Integer incx,  Complex *y, Integer incy,
              Complex *a, Integer tda)
```

$A \leftarrow \alpha x y^H + y(\alpha x)^H + A$, A is n by n Hermitian in packed form

```
void  chpr2 (MatrixTriangle uplo, Integer n, Scomplex alpha,
             Scomplex *x, Integer incx, Scomplex *y, Integer incy,
             Scomplex *ap)
void  zhpr2 (MatrixTriangle uplo, Integer n,  Complex alpha,
              Complex *x, Integer incx,  Complex *y, Integer incy,
              Complex *ap)
```

$A \leftarrow \alpha x x^T + A$, A is n by n symmetric

```
void   ssyr (MatrixTriangle uplo, Integer n,    float alpha,
                float *x, Integer incx,    float *a, Integer tda)
void   dsyr (MatrixTriangle uplo, Integer n,   double alpha,
               double *x, Integer incx,   double *a, Integer tda)
```

$A \leftarrow \alpha x x^T + A$, A is n by n symmetric in packed form

```
void   sspr (MatrixTriangle uplo, Integer n,    float alpha,
                float *x, Integer incx,    float *ap)
void   dspr (MatrixTriangle uplo, Integer n,   double alpha,
               double *x, Integer incx,   double *ap)
```

$A \leftarrow \alpha x y^T + \alpha y x^T + A$, A is n by n symmetric

```
void  ssyr2 (MatrixTriangle uplo, Integer n,    float alpha,
                float *x, Integer incx,    float *y, Integer incy,
                float *a, Integer tda)
void  dsyr2 (MatrixTriangle uplo, Integer n,   double alpha,
               double *x, Integer incx,   double *y, Integer incy,
               double *a, Integer tda)
```

$A \leftarrow \alpha x y^T + \alpha y x^T + A$, A is n by n symmetric in packed form

```
void  sspr2 (MatrixTriangle uplo, Integer n,    float alpha,
                float *x, Integer incx,    float *y, Integer incy,
                float *ap)
void  dspr2 (MatrixTriangle uplo, Integer n,   double alpha,
               double *x, Integer incx,   double *y, Integer incy,
               double *ap)
```

Level 3 BLAS in C

$C \leftarrow \alpha op(A)op(B) + \beta C$, $op(X) = X, X^T, X^H$, C is m by n

```
void  sgemm (MatrixTranspose transa, MatrixTranspose transb,
             Integer m, Integer n, Integer k,    float alpha,
                float *a, Integer tda,    float *b, Integer tdb,
                float beta,    float *c, Integer tdc)
void  dgemm (MatrixTranspose transa, MatrixTranspose transb,
             Integer m, Integer n, Integer k,   double alpha,
               double *a, Integer tda,   double *b, Integer tdb,
               double beta,   double *c, Integer tdc)
void  cgemm (MatrixTranspose transa, MatrixTranspose transb,
             Integer m, Integer n, Integer k, Scomplex alpha,
             Scomplex *a, Integer tda, Scomplex *b, Integer tdb,
             Scomplex beta, Scomplex *c, Integer tdc)
void  zgemm (MatrixTranspose transa, MatrixTranspose transb,
             Integer m, Integer n, Integer k,  Complex alpha,
              Complex *a, Integer tda,  Complex *b, Integer tdb,
              Complex beta,  Complex *c, Integer tdc)
```

$C \leftarrow \alpha AB + \beta C$, $C \leftarrow \alpha BA + \beta C$, C is m by n, A is m by m symmetric

```
void  ssymm (OperationSide side, MatrixTriangle uplo, Integer m,
             Integer n,    float alpha,    float *a, Integer tda,
                float *b, Integer tdb,    float beta,
                float *c, Integer tdc)
void  dsymm (OperationSide side, MatrixTriangle uplo, Integer m,
             Integer n,   double alpha,   double *a, Integer tda,
               double *b, Integer tdb,   double beta,
               double *c, Integer tdc)
void  csymm (OperationSide side, MatrixTriangle uplo, Integer m,
             Integer n, Scomplex alpha, Scomplex *a, Integer tda,
             Scomplex *b, Integer tdb, Scomplex beta,
             Scomplex *c, Integer tdc)
void  zsymm (OperationSide side, MatrixTriangle uplo, Integer m,
             Integer n,  Complex alpha,  Complex *a, Integer tda,
              Complex *b, Integer tdb,  Complex beta,
              Complex *c, Integer tdc)
```

$C \leftarrow \alpha AB + \beta C$, $C \leftarrow \alpha BA + \beta C$, C is m by n, A is m by m Hermitian

```
void  chemm (OperationSide side, MatrixTriangle uplo, Integer m,
             Integer n, Scomplex alpha, Scomplex *a, Integer tda,
             Scomplex *b, Integer tdb, Scomplex beta,
             Scomplex *c, Integer tdc)
void  zhemm (OperationSide side, MatrixTriangle uplo, Integer m,
             Integer n,  Complex alpha,  Complex *a, Integer tda,
              Complex *b, Integer tdb,  Complex beta,
              Complex *c, Integer tdc)
```

$C \leftarrow \alpha AA^T + \beta C$, $C \leftarrow \alpha A^T A + \beta C$, C is n by n

```
void  ssyrk (MatrixTriangle uplo, MatrixTranspose trans, Integer n,
             Integer k,    float alpha,    float *a, Integer tda,
                float beta,    float *c, Integer tdc)
void  dsyrk (MatrixTriangle uplo, MatrixTranspose trans, Integer n,
             Integer k,   double alpha,   double *a, Integer tda,
               double beta,   double *c, Integer tdc)
void  csyrk (MatrixTriangle uplo, MatrixTranspose trans, Integer n,
             Integer k, Scomplex alpha, Scomplex *a, Integer tda,
             Scomplex beta, Scomplex *c, Integer tdc)
void  zsyrk (MatrixTriangle uplo, MatrixTranspose trans, Integer n,
             Integer k,  Complex alpha,  Complex *a, Integer tda,
              Complex beta,  Complex *c, Integer tdc)
```

$C \leftarrow \alpha AA^H + \beta C$, $C \leftarrow \alpha A^H A + \beta C$, C is n by n

```
void  cherk (MatrixTriangle uplo, MatrixTranspose trans, Integer n,
             Integer k, Scomplex alpha, Scomplex *a, Integer tda,
             Scomplex beta, Scomplex *c, Integer tdc)
void  zherk (MatrixTriangle uplo, MatrixTranspose trans, Integer n,
             Integer k,  Complex alpha,  Complex *a, Integer tda,
              Complex beta,  Complex *c, Integer tdc)
```

$C \leftarrow \alpha AB^H + \alpha BA^H + \beta C$, $C \leftarrow \alpha A^H B + \alpha B^H A + \beta C$, C is n by n

```
void cher2k (MatrixTriangle uplo, MatrixTranspose trans, Integer n,
             Integer k, Scomplex alpha, Scomplex *a, Integer tda,
             Scomplex *b, Integer tdb, Scomplex beta,
             Scomplex *c, Integer tdc)
void zher2k (MatrixTriangle uplo, MatrixTranspose trans, Integer n,
             Integer k,  Complex alpha,  Complex *a, Integer tda,
              Complex *b, Integer tdb,  Complex beta,
              Complex *c, Integer tdc)
```

$C \leftarrow \alpha AB^T + \alpha BA^T + \beta C$, $C \leftarrow \alpha A^T B + \alpha B^T A + \beta C$, C is n by n

```
void ssyr2k (MatrixTriangle uplo, MatrixTranspose trans, Integer n,
             Integer k,    float alpha,    float *a, Integer tda,
                float *b, Integer tdb,    float beta,
                float *c, Integer tdc)
void dsyr2k (MatrixTriangle uplo, MatrixTranspose trans, Integer n,
             Integer k,   double alpha,   double *a, Integer tda,
               double *b, Integer tdb,   double beta,
               double *c, Integer tdc)
void csyr2k (MatrixTriangle uplo, MatrixTranspose trans, Integer n,
             Integer k, Scomplex alpha, Scomplex *a, Integer tda,
             Scomplex *b, Integer tdb, Scomplex beta,
             Scomplex *c, Integer tdc)
```

```
void zsyr2k (MatrixTriangle uplo, MatrixTranspose trans, Integer n,
             Integer k,  Complex alpha,  Complex *a, Integer tda,
              Complex *b, Integer tdb,  Complex beta,
              Complex *c, Integer tdc)
```

$B \leftarrow \alpha op(A)B$, $B \leftarrow \alpha Bop(A)$, $op(A) = A, A^T, A^H$, B is m by n

```
void  strmm (MatrixTriangle side, MatrixTriangle uplo,
             MatrixTranspose transa, MatrixUnitTriangular diag,
             Integer m, Integer n,    float alpha,
                float *a, Integer tda,    float *b, Integer tdb)
void  dtrmm (MatrixTriangle side, MatrixTriangle uplo,
             MatrixTranspose transa, MatrixUnitTriangular diag,
             Integer m, Integer n,   double alpha,
               double *a, Integer tda,   double *b, Integer tdb)
void  ctrmm (MatrixTriangle side, MatrixTriangle uplo,
             MatrixTranspose transa, MatrixUnitTriangular diag,
             Integer m, Integer n, Scomplex alpha,
             Scomplex *a, Integer tda, Scomplex *b, Integer tdb)
void  ztrmm (MatrixTriangle side, MatrixTriangle uplo,
             MatrixTranspose transa, MatrixUnitTriangular diag,
             Integer m, Integer n,  Complex alpha,
              Complex *a, Integer tda,  Complex *b, Integer tdb)
```

$B \leftarrow \alpha op(A^{-1})B$, $B \leftarrow \alpha Bop(A^{-1})$, $op(A) = A, A^T, A^H$, B is m by n

```
void  strsm (OperationSide side, MatrixTriangle uplo,
             MatrixTranspose transa, MatrixUnitTriangular diag,
             Integer m, Integer n,    float alpha,
                float *a, Integer tda,    float *b, Integer tdb)
void  dtrsm (OperationSide side, MatrixTriangle uplo,
             MatrixTranspose transa, MatrixUnitTriangular diag,
             Integer m, Integer n,   double alpha,
               double *a, Integer tda,   double *b, Integer tdb)
void  ctrsm (OperationSide side, MatrixTriangle uplo,
             MatrixTranspose transa, MatrixUnitTriangular diag,
             Integer m, Integer n, Scomplex alpha,
             Scomplex *a, Integer tda, Scomplex *b, Integer tdb)
void  ztrsm (OperationSide side, MatrixTriangle uplo,
             MatrixTranspose transa, MatrixUnitTriangular diag,
             Integer m, Integer n,  Complex alpha,
              Complex *a, Integer tda,  Complex *b, Integer tdb)
```