**Matrix Storage Format in BLAS/LAPACK**

There are three levels of routines involved in BLAS. Level 1 is mostly focusing on vectors, level 2 and 3 are involved with matrices, thus there is a need to talk about matrix storage in computer memories and how they were handled in C/C++ and FORTRAN.

Here is a table of all possible matrix formats and their storage is described later.

* GE - GEneral
* GB - General Band
* SY - SYmmetric
* SB - Symmetric Band
* SP - Symmetric Packed
* HE - HErmitian
* HB - Hermitian Band
* HP - Hermitian Packed
* TR - TRiangular
* TB - Triangular Band
* TP - Triangular Packed

# Matrix Storage Modes and Examples

## General Matrix

General format is basically the usual way matrices are saved during programming. Normally for C/C++, a matrix is saved in **row-major** order; while for FORTRAN, **column-major** order is used. For example, Matrix A is given as below:

[8 2 2 9

9 1 4 4

3 5 4 5]

The column-major order to store it is as follows:

[8 9 3 2 1 5 2 4 4 9 4 5]

The row-major order to store it is as follows:

[8 2 2 9 9 1 4 4 3 5 4 5]

Since BLAS was originally written in FORTRAN, thus when programming in C/C++, to make sure the matrices are correctly passed to BLAS, you should store the matrix in column-major order. This is actually in accordance with above matrix storage strategy, not contradictory.

An example matrix is defined below. lda = 10, M = 4, N = 3. To use A in C/C++ language, use &A[1] in the routines.

| . . . |

| 1.0 2.0 3.0 |

| 2.0 2.0 4.0 |

| 3.0 2.0 2.0 |

| 4.0 2.0 1.0 |

A = | . . . |

| . . . |

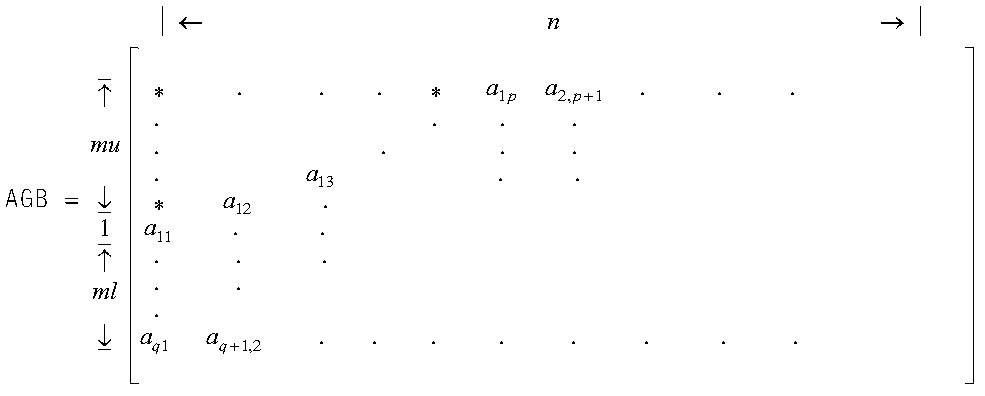
| . . . |

| . . . |

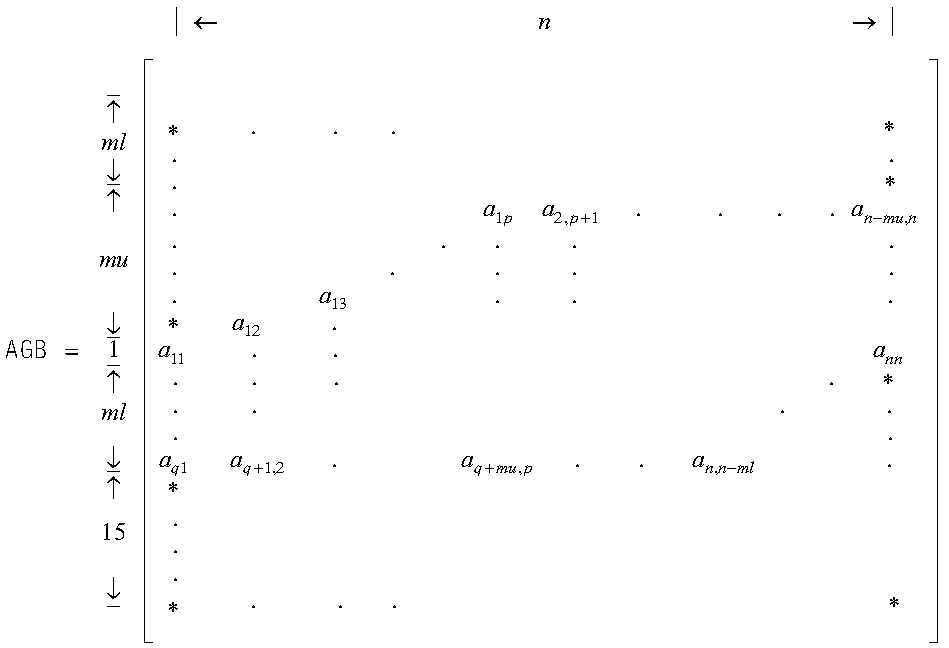
| . . . |

## General Band Matrix

BLAS adopted a memory-saving technique to store band matrices. It can be used for both square and rectangular matrices. The band matrix A is saved in the form below:



There are some differences between BLAS general-band storage and other general-band storage seen in literatures. The other general-band storage only works for square matrix and defined as:



An example is given below. A is a band matrix of size 5x4. We can determine that ml=3, mu=2.

| 1.0 1.0 1.0 0.0 |

| 2.0 2.0 2.0 2.0 |

A = | 3.0 3.0 3.0 3.0 |

| 4.0 4.0 4.0 4.0 |

| 0.0 5.0 5.0 5.0 |

If lda=8, when storing it, it can be written as below.

| . . 1.0 2.0 |

| . 1.0 2.0 3.0 |

| 1.0 2.0 3.0 4.0 |

A = | 2.0 3.0 4.0 5.0 |

| 3.0 4.0 5.0 . |

| 4.0 5.0 . . |

| . . . . |

| . . . . |

## Symmetric Matrix

For symmetric matrix, BLAS can store only half of the matrix to save memory. It can be done in either upper or lower storage mode.

An example is given below. A is a symmetric matrix of size 3x3.

| 8.0 4.0 2.0 |

A = | 4.0 6.0 7.0 |

| 2.0 7.0 3.0 |

We can define as follows using lower storage mode:

| 8.0 . . |

A = | 4.0 6.0 . |

| 2.0 7.0 3.0 |

## Symmetric Band Matrix

We can further exploit the band structure of symmetric matrix to save memory.

An example is given below. A is a symmetric band matrix of size 7x7. We can determine that k=3. Here k is the half band width of the matrix A. lda = 5.

| 1.0 1.0 1.0 1.0 0.0 0.0 0.0 |

| 1.0 2.0 2.0 2.0 2.0 0.0 0.0 |

| 1.0 2.0 3.0 3.0 3.0 3.0 0.0 |

A = | 1.0 2.0 3.0 4.0 4.0 4.0 4.0 |

| 0.0 2.0 3.0 4.0 5.0 5.0 5.0 |

| 0.0 0.0 3.0 4.0 5.0 6.0 6.0 |

| 0.0 0.0 0.0 4.0 5.0 6.0 7.0 |

We can define it as follows in upper-hand-packed storage mode:

| . . . 1.0 2.0 3.0 4.0 |

| . . 1.0 2.0 3.0 4.0 5.0 |

A = | . 1.0 2.0 3.0 4.0 5.0 6.0 |

| 1.0 2.0 3.0 4.0 5.0 6.0 7.0 |

| . . . . . . . |

## Symmetric Matrix in *Packed Mode*

Packed mode basically tries to avoid writing the whole matrix entries, but only write necessary ones in an array.

An example is given below. A is a 3X3 symmetric matrix.

| 8.0 4.0 2.0 |

A = | 4.0 6.0 7.0 |

| 2.0 7.0 3.0 |

We can define it in lower-packed storage mode:

AP = (8.0, 4.0, 2.0, 6.0, 7.0, 3.0)

## Hermitian Matrix

Hermitian matrix basically refers to a complex square matrix that is equal to its own conjugate transpose. It needs to be distinguished from above symmetric complex matrix. Its property can help save memory by only storing half of the matrix.

An example is given below. A is a 3X3 Hermitian matrix.

| (1.0, . ) . . |

A = | (3.0, -5.0) (7.0, . ) . |

| (2.0, 3.0) (4.0, 8.0) (6.0, . ) |

## Hermitian Band Matrix

The band structure of Hermitian matrix can of course be exploited to save memory as well.

An example is given below. A is a 7X7 Hermitian matrix. We can identify that k = 3.

| (1.0, 0.0) (1.0, 1.0) (1.0, 1.0) (1.0, 1.0) (0.0, 0.0) (0.0, 0.0) (0.0, 0.0) |

| (1.0, -1.0) (2.0, 0.0) (2.0, 2.0) (2.0, 2.0) (2.0, 2.0) (0.0, 0.0) (0.0, 0.0) |

| (1.0, -1.0) (2.0, -2.0) (3.0, 0.0) (3.0, 3.0) (3.0, 3.0) (3.0, 3.0) (0.0, 0.0) |

| (1.0, -1.0) (2.0, -2.0) (3.0, -3.0) (4.0, 0.0) (4.0, 4.0) (4.0, 4.0) (4.0, 4.0) |

| (0.0, 0.0) (2.0, -2.0) (3.0, -3.0) (4.0, -4.0) (5.0, 0.0) (5.0, 5.0) (5.0, 5.0) |

| (0.0, 0.0) (0.0, 0.0) (3.0, -3.0) (4.0, -4.0) (5.0, -5.0) (6.0, 0.0) (6.0, 6.0) |

| (0.0, 0.0) (0.0, 0.0) (0.0, 0.0) (4.0, -4.0) (5.0, -5.0) (6.0, -6.0) (7.0, 0.0) |

We can define it in lower-band-packed storage mode as follows:

| (1.0, . ) (2.0, . ) (3.0, . ) (4.0, . ) (5.0, . ) (6.0, . ) (7.0, . ) |

| (1.0, 1.0) (2.0, 2.0) (3.0, 3.0) (4.0, 4.0) (5.0, 5.0) (6.0, 6.0) . |

A = | (1.0, 1.0) (2.0, 2.0) (3.0, 3.0) (4.0, 4.0) (5.0, 5.0) . . |

| (1.0, 1.0) (2.0, 2.0) (3.0, 3.0) (4.0, 4.0) . . . |

| . . . . . . . |

## Hermitian Matrix in *Packed Mode*

Packed mode basically means to store the matrix as an array. The

An example is given below.

| (1.0, 0.0) (3.0, 5.0) (2.0, -3.0) |

A = | (3.0, -5.0) (7.0, 0.0) (4.0, -8.0) |

| (2.0, 3.0) (4.0, 8.0) (6.0, 0.0) |

We can define it as follows in lower-packed storage mode:

AP = ((1.0, . ), (3.0, -5.0), (2.0, 3.0), (7.0, . ), (4.0, 8.0), (6.0, . ))

## Triangular Matrix

Triangular matrix means only half of the matrix is filled with values, while the other entries are not used. It can be defined with unit or non-unit diagonal terms. For unit triangular matrix, the diagonal elements adopt a value of 1.0, and don’t need to be defined.

For example, A is a lower triangular matrix. lda = 4.

| 1.0 . . . |

| 1.0 1.0 . . |

A = | 2.0 3.0 1.0 . |

| 3.0 4.0 3.0 1.0 |

We can define it as follows in lower-triangular storage mode (unit diagonal):

| . . . . |

A = | 1.0 . . . |

| 2.0 3.0 . . |

| 3.0 4.0 3.0 . |

## Triangular Band Matrix

Band matrix for triangular matrix works the same as above.

An example is given below. lda = 5, k = 3.

| 1.0 1.0 1.0 1.0 0.0 0.0 0.0 |

| 0.0 2.0 2.0 2.0 2.0 0.0 0.0 |

| 0.0 0.0 3.0 3.0 3.0 3.0 0.0 |

| 0.0 0.0 0.0 4.0 4.0 4.0 4.0 |

| 0.0 0.0 0.0 0.0 5.0 5.0 5.0 |

| 0.0 0.0 0.0 0.0 0.0 6.0 6.0 |

| 0.0 0.0 0.0 0.0 0.0 0.0 7.0 |

We can define it as follows in upper-triangular-band-packed storage mode:

| . . . 1.0 2.0 3.0 4.0 |

| . . 1.0 2.0 3.0 4.0 5.0 |

A = | . 1.0 2.0 3.0 4.0 5.0 6.0 |

| 1.0 2.0 3.0 4.0 5.0 6.0 7.0 |

| . . . . . . . |

## Triangular Matrix in *Packed Mode*

Same as above, packed storage mode can also be used with triangular matrix.

An example is given below for a upper triangular matrix.

| 1.0 2.0 3.0 2.0 |

| . 2.0 2.0 5.0 |

| . . 3.0 3.0 |

| . . . 1.0 |

We can define it as follows in upper-triangular-packed storage mode (non-unit diagonal):

AP = (1.0, 2.0, 2.0, 3.0, 2.0, 3.0, 2.0, 5.0, 3.0, 1.0)

**Below some examples are given regarding how to use BLAS level-2 functions.**

# Matrix-Vector Product

The following BLAS level 2 routines are relevant to (Matrix-Vector Product for a Matrix, Its Transpose, or Its Conjugate Transpose):

* SGEMV, DGEMV, CGEMV, ZGEMV
* SGBMV, DGBMV, CGBMV, ZGBMV
* SSYMV, DSYMV, CHEMV, ZHEMV
* SSBMV, DSBMV, CHBMV, ZHBMV
* SSPMV, DSPMV, CHPMV, ZHPMV
* STRMV, DTRMV, CTRMV, ZTRMV
* STBMV, DTBMV, CTBMV, ZTBMV
* STPMV, DTPMV, CTPMV, ZTPMV

SxxMV and DxxMV works for either a real matrix or its transpose, using the scalars α and β, vectors x and y, and matrix A or its transpose:

y ← βy +αAx

y ← βy +αATx

CxxMV and ZxxMV works for either a complex matrix, its transpose, or its conjugate transpose, using the scalars α and β, vectors x and y, and matrix A, its transpose, or its conjugate transpose:

y ← β y +αAx

y ← βy +αATx

y ← βy +αAHx

The syntax to use these functions in different languages are summarized below.

**For general matrix:**

| **Language** | **Syntax** |
| --- | --- |
| Fortran | CALL SGEMV | DGEMV | CGEMV | ZGEMV (*transa*, *m*, *n*, *alpha*, *a*, *lda*, *x*, *incx*, *beta*, *y*, *incy*) |
| C and C++ | sgemv | dgemv | cgemv | zgemv (*transa*, *m*, *n*, *alpha*, *a*, *lda*, *x*, *incx*, *beta*, *y*, *incy*); |
| CBLAS | cblas\_sgemv | cblas\_dgemv | cblas\_cgemv | cblas\_zgemv (*cblas\_layout*, *cblas\_transa*, *m*, *n*, *alpha*, *a*, *lda*, *x*, *incx*, *beta*, *y*, *incy*); |

**For general band matrix:**

| **Language** | **Syntax** |
| --- | --- |
| Fortran | CALL SGBMV | DGBMV | CGBMV | ZGBMV  *(transa, m, n, ml, mu, alpha, a, lda, x, incx, beta, y, incy)* |
| C and C++ | sgbmv | dgbmv | cgbmv | zgbmv  (*transa, m, n, ml, mu, alpha, a, lda, x, incx, beta, y, incy*); |
| CBLAS | cblas\_sgbmv | cblas\_dgbmv |cblas\_cgbmv | cblas\_zgbmv  (*cblas\_layout, cblas\_transa, m, n, ml, mu, alpha, a, lda, x, incx, beta, y, incy*); |

**For symmetric or symmetric packed storage:**

| **Language** | **Syntax** |
| --- | --- |
| Fortran | CALL SSPMV | DSPMV | CHPMV | ZHPMV (*uplo*, *n*, *alpha*, *ap*, *x*, *incx*, *beta*, *y*, *incy*)  CALL SSYMV | DSYMV | CHEMV | ZHEMV (*uplo*, *n*, *alpha*, *a*, *lda*, *x*, *incx*, *beta*, *y*, *incy*)  CALL SSLMX | DSLMX (*n*, *alpha*, *ap*, *x*, *incx*, *y*, *incy*) |
| C and C++ | sspmv | dspmv | chpmv | zhpmv (*uplo*, *n*, *alpha*, *ap*, *x*, *incx*, *beta*, *y*, *incy*);  ssymv | dsymv | chemv | zhemv (*uplo*, *n*, *alpha*, *a*, *lda*, *x*, *incx*, *beta*, *y*, *incy*);  sslmx | dslmx (*n*, *alpha*, *ap*, *x*, *incx*, *y*, *incy*); |
| CBLAS | cblas\_sspmv | cblas\_dspmv | cblas\_chpmv | cblas\_zhpmv (*cblas\_layout*, *cblas\_uplo*, *n*, *alpha*, *ap*, *x*, *incx*, *beta*, *y*, *incy*);  cblas\_ssymv | cblas\_dsymv | cblas\_chemv | cblas\_zhemv (*cblas\_layout*, *cblas\_uplo*, *n*, *alpha*, *a*, *lda*, *x*, *incx*, *beta*, *y*, *incy*); |

**For symmetric band matrix:**

| **Language** | **Syntax** |
| --- | --- |
| **Fortran** | CALL SSBMV | DSBMV | CHBMV | ZHBMV (*uplo*, *n*, *k*, *alpha*, *a*, *lda*, *x*, *incx*, *beta*, *y*, *incy*) |
| **C and C++** | ssbmv | dsbmv | chbmv | zhbmv (*uplo*, *n*, *k*, *alpha*, *a*, *lda*, *x*, *incx*, *beta*, *y*, *incy*); |
| **CBLAS** | cblas\_ssbmv | cblas\_dsbmv | cblas\_chbmv | cblas\_zhbmv (*cblas\_layout*, *cblas\_uplo*, *n*, *k*, *alpha*, *a*, *lda*, *x*, *incx*, *beta*, *y*, *incy*); |

**For triangular matrix and its packed storage:**

| **Language** | **Syntax** |
| --- | --- |
| **Fortran** | CALL STRMV | DTRMV | CTRMV | ZTRMV (*uplo*, *transa*, *diag*, *n*, *a*, *lda*, *x*, *incx*)  CALL STPMV | DTPMV | CTPMV | ZTPMV (*uplo*, *transa*, *diag*, *n*, *ap*, *x*, *incx*) |
| **C and C++** | strmv | dtrmv | ctrmv | ztrmv (*uplo*, *transa*, *diag*, *n*, *a*, *lda*, *x*, *incx*);  stpmv | dtpmv | ctpmv | ztpmv (*uplo*, *transa*, *diag*, *n*, *ap*, *x*, *incx*); |
| **CBLAS** | cblas\_strmv | cblas\_dtrmv | cblas\_ctrmv | cblas\_ztrmv (*cblas\_layout*, *cblas\_uplo*, *cblas\_transa*, *cblas\_diag*, *n*, *a*, *lda*, *x*, *incx*);  cblas\_stpmv | cblas\_dtpmv | cblas\_ctpmv | cblas\_ztpmv (*cblas\_layout*, *cblas\_uplo*, *cblas\_transa*, *cblas\_diag*, *n*, *ap*, *x*, *incx*); |

**For triangular band matrix:**

| **Language** | **Syntax** |
| --- | --- |
| **Fortran** | CALL STBMV | DTBMV | CTBMV | ZTBMV (*uplo*, *transa*, *diag*, *n*, *k*, *a*, *lda*, *x*, *incx*) |
| **C and C++** | stbmv | dtbmv | ctbmv | ztbmv (*uplo*, *transa*, *diag*, *n*, *k*, *a*, *lda*, *x*, *incx*); |
| **CBLAS** | cblas\_stbmv | cblas\_dtbmv | cblas\_ctbmv | cblas\_ztbmv (*cblas\_layout*, *cblas\_uplo*, *cblas\_transa*, *cblas\_diag*, *n*, *k*, *a*, *lda*, *x*, *incx*); |

Some arguments are explained here:

cblas\_layout

indicates whether the input matrices are stored in row-major order or column-major order, where:

If cblas\_layout = CblasRowMajor, the matrices are stored in row major order.

If cblas\_layout = CblasColMajor, the matrices are stored in column major order.

Specified as an object of enumerated type CBLAS\_LAYOUT.

transa

indicates the form of matrix A to use in the computation, where:

If transa = 'N', A is used in the computation.

If transa = 'T', AT is used in the computation.

If transa = 'C', AH is used in the computation.

Specified as a single character. It must be 'N', 'T', or 'C'.

cblas\_transa

indicates the form of matrix A to use in the computation, where:

If cblas\_transa = CblasNoTrans, A is used in the computation.

If cblas\_transa = CblasTrans, AT is used in the computation.

If cblas\_transa = CblasConjTrans, AH is used in the computation.

Specified as an object of enumerated type CBLAS\_TRANSPOSE.

diag

indicates the characteristics of the diagonal of matrix A, where:

If diag = 'U', A is a unit triangular matrix.

If diag = 'N', A is not a unit triangular matrix.

Specified as: a single character. It must be 'U' or 'N'.

cblas\_diag

indicates the characteristics of the diagonal of matrix A, where:

If diag = CblasUnit, A is a unit triangular matrix.

If diag = CblasNonUnit A is not a unit triangular matrix.

Specified as: an object of enumerated type CBLAS\_DIAG.

uplo

indicates whether matrix A is an upper or lower triangular matrix, where:

If uplo = 'U', A is an upper triangular matrix.

If uplo = 'L', A is a lower triangular matrix.

Specified as: a single character. It must be 'U' or 'L'.

cblas\_uplo

indicates whether matrix A is an upper or lower triangular matrix, where:

If cblas\_uplo = CblasUpper, A is an upper triangular matrix.

If cblas\_uplo = CblasLower, A is a lower triangular matrix.

Specified as: an object of enumerated type CBLAS\_UPLO.

m, n

is the number of rows and columns in matrix A

Specified as an integer; 0 ≤ m ≤ lda.

Specified as an integer; n ≥ 0.

a

is the m by n matrix A, where:

For SGEMV, DGEMV, CGEMV, and ZGEMV:

If transa = 'N', A is used in the computation.

If transa = 'T', AT is used in the computation.

If transa = 'C', AH is used in the computation.

Specified as an lda by (at least) n array, containing numbers of the data type.

ap

is the upper or lower triangular matrix A of order n, stored in upper- or lower-triangular-packed storage mode, respectively.

Specified as: a one-dimensional array of (at least) length n(n+1)/2.

lda

is the leading dimension of the array specified for a.

Specified as an integer; lda > 0 and lda ≥ m.

x, y

is the vector x and y, where:

incx

is the stride for vector x.

Specified as: an integer; It can have any value.

incy

is the stride for vector y.

Specified as an integer; incy > 0 or incy < 0.

ml

is the lower band width ml of the matrix A.

Specified as: an integer; ml ≥ 0.

mu

is the upper band width mu of the matrix A.

Specified as: an integer; mu ≥ 0.

**Notes**

* For SGBMV and DGBMV, if you specify 'C' for the transa argument, it is interpreted as though you specified 'T'.
* All subroutines accept lowercase letters for the transa argument.
* Vector y must have no common elements with matrix A or vector x; otherwise, results are unpredictable. See Vector concepts.
* To achieve optimal performance, use lda = mu+ml+1.
* In all the computations, general band matrix A is stored in its untransposed form in an array, using BLAS-general-band storage mode.

Some examples demonstrating these routines are given. The testing code of the examples are located at **test-blas/level-2** in the repository.

**SGEMV Example:**

TRANSA M N ALPHA A LDA X INCX BETA Y INCY

| | | | | | | | | | |

CALL SGEMV( 'N' , 4 , 3 , 1.0 , A(1,0) , 10 , X , 1 , 1.0 , Y , 2 )

| . . . |

| 1.0 2.0 3.0 |

| 2.0 2.0 4.0 |

| 3.0 2.0 2.0 |

| 4.0 2.0 1.0 |

A = | . . . |

| . . . |

| . . . |

| . . . |

| . . . |

X = (3.0, 2.0, 1.0)

Y = (4.0, . , 5.0, . , 2.0, . , 3.0)

Output:

Y = (14.0, . , 19.0, . , 17.0, . , 20.0)

**CGBMV Example:**

TRANSA M N ML MU ALPHA A LDA X INCX BETA Y INCY

| | | | | | | | | | | | |

CALL CGBMV( 'C' , 5 , 4 , 3 , 2 , ALPHA , A , 8 , X , 1 , BETA , Y , 2 )

A =

| (1.0, 1.0) (1.0, 1.0) (1.0, 1.0) (0.0, 0.0) |

| (2.0, 2.0) (2.0, 2.0) (2.0, 2.0) (2.0, 2.0) |

| (3.0, 3.0) (3.0, 3.0) (3.0, 3.0) (3.0, 3.0) |

| (4.0, 4.0) (4.0, 4.0) (4.0, 4.0) (4.0, 4.0) |

| (0.0, 0.0) (5.0, 5.0) (5.0, 5.0) (0.0, 0.0) |

| . . (1.0, 1.0) (2.0, 2.0) |

| . (1.0, 1.0) (2.0, 2.0) (3.0, 3.0) |

| (1.0, 1.0) (2.0, 2.0) (3.0, 3.0) (4.0, 4.0) |

A = | (2.0, 2.0) (3.0, 3.0) (4.0, 4.0) (5.0, 5.0) |

| (3.0, 3.0) (4.0, 4.0) (5.0, 5.0) . |

| (4.0, 4.0) (5.0, 5.0) . . |

| . . . . |

| . . . . |

X = ((1.0, 2.0), (2.0, 3.0), (3.0, 4.0), (4.0, 5.0), (5.0, 6.0))

ALPHA = (1.0, 1.0)

BETA = (10.0, 0.0)

Y = ((1.0, 2.0), . , (2.0, 3.0), . , (3.0, 4.0), . ,(4.0, 5.0), . )

Output:

Y = ((70.0, 100.0), . , (130.0, 170.0), . ,

(140.0, 180.0), . , (148.0, 186.0), . )

**SSPMV Example:**

| 8.0 4.0 2.0 |

| 4.0 6.0 7.0 |

| 2.0 7.0 3.0 |

UPLO N ALPHA AP X INCX BETA Y INCY

| | | | | | | | |

CALL SSPMV( 'L' , 3 , 1.0 , AP , X , 1 , 1.0 , Y , 2 )

AP = (8.0, 4.0, 2.0, 6.0, 7.0, 3.0)

UPLO N ALPHA AP X INCX BETA Y INCY

| | | | | | | | |

CALL SSPMV( 'U' , 3 , 1.0 , AP , X , -2 , 2.0 , Y , 1 )

AP = (8.0, 4.0, 6.0, 2.0, 7.0, 3.0)

**SGBMV Example:**

| 1.0 1.0 1.0 0.0 |

| 2.0 2.0 2.0 2.0 |

| 3.0 3.0 3.0 3.0 |

| 4.0 4.0 4.0 4.0 |

| 0.0 5.0 5.0 5.0 |

TRANSA M N KL KU ALPHA A LDA X INCX BETA Y INCY

| | | | | | | | | | | | |

CALL SGBMV( 'N' , 5 , 4 , 3 , 2 , 2.0 , A , 8 , X , 1 , 10.0 , Y , 2 )

| . . 1.0 2.0 |

| . 1.0 2.0 3.0 |

| 1.0 2.0 3.0 4.0 |

A = | 2.0 3.0 4.0 5.0 |

| 3.0 4.0 5.0 . |

| 4.0 5.0 . . |

| . . . . |

| . . . . |

**SSYMV Example:**

UPLO N ALPHA A LDA X INCX BETA Y INCY

| | | | | | | | | |

CALL SSYMV( 'L' , 3 , 1.0 , A , 3 , X , 1 , 1.0 , Y , 2 )

| 8.0 . . |

A = | 4.0 6.0 . |

| 2.0 7.0 3.0 |

UPLO N ALPHA A LDA X INCX BETA Y INCY

| | | | | | | | | |

CALL SSYMV( 'U' , 3 , 1.0 , A , 4 , X , -2 , 2.0 , Y , 1 )

| 8.0 4.0 2.0 |

A = | . 6.0 7.0 |

| . . 3.0 |

| . . . |

**SSBMV Example:**

| 1.0 1.0 1.0 1.0 1.0 |

| 1.0 2.0 2.0 2.0 2.0 |

| 1.0 2.0 3.0 3.0 3.0 |

| 1.0 2.0 3.0 4.0 4.0 |

| 1.0 2.0 3.0 4.0 5.0 |

UPLO N K ALPHA A LDA X INCX BETA Y INCY

| | | | | | | | | | |

CALL SSBMV( 'U' , 5 , 5 , 2.0 , A , 7 , X , 1 , 10.0 , Y , 2 )

| . . . . . |

| . . . . 1.0 |

| . . . 1.0 2.0 |

A = | . . 1.0 2.0 3.0 |

| . 1.0 2.0 3.0 4.0 |

| 1.0 2.0 3.0 4.0 5.0 |

| . . . . . |

**Example on Band-packed Storage:**

Lower-Band-Packed Storage Mode

| 11 21 31 0 0 0 |

| 21 22 32 42 0 0 |

| 31 32 33 43 53 0 |

| 0 42 43 44 54 64 |

| 0 0 53 54 55 65 |

| 0 0 0 64 65 66 |

Saving as:

| 11 22 33 44 55 66 |

ASB = | 21 32 43 54 65 \* |

| 31 42 53 64 \* \* |

Upper-Band-Packed Storage Mode

| 11 12 13 14 0 0 |

| 12 22 23 24 25 0 |

| 13 23 33 34 35 36 |

| 14 24 34 44 45 46 |

| 0 25 35 45 55 56 |

| 0 0 36 46 56 66 |

Saving as:

| \* \* \* 14 25 36 |

ASB = | \* \* 13 24 35 46 |

| \* 12 23 34 45 56 |

| 11 22 33 44 55 66 |

**STRMV Example:**

| 1.0 . . . |

| 1.0 1.0 . . |

| 2.0 3.0 1.0 . |

| 3.0 4.0 3.0 1.0 |

UPLO TRANSA DIAG N A LDA X INCX

| | | | | | | |

CALL STRMV( 'L' , 'N' , 'U' , 4 , A , 4 , X , 1 )

| . . . . |

A = | 1.0 . . . |

| 2.0 3.0 . . |

| 3.0 4.0 3.0 . |

**STRMV Example 2:**

| 1.0 2.0 3.0 2.0 |

| . 1.0 2.0 5.0 |

| . . 1.0 3.0 |

| . . . 1.0 |

UPLO TRANSA DIAG N A LDA X INCX

| | | | | | | |

CALL STRMV( 'U' , 'T' , 'U' , 4 , A , 4 , X , 1 )

| . 2.0 3.0 2.0 |

A = | . . 2.0 5.0 |

| . . . 3.0 |

| . . . . |

**STPMV Example:**

Matrix **A** is a real 4 by 4 upper triangular matrix that is not unit triangular, stored in upper-triangular-packed storage mode.

| 1.0 . . . |

| 1.0 1.0 . . |

| 2.0 3.0 1.0 . |

| 3.0 4.0 3.0 1.0 |

UPLO TRANSA DIAG N AP X INCX

| | | | | | |

CALL STPMV( 'L' , 'N' , 'U' , 4 , AP , X , 1 )

AP = ( . , 1.0, 2.0, 3.0, . , 3.0, 4.0, . , 3.0, . )

| 1.0 2.0 3.0 2.0 |

| . 2.0 2.0 5.0 |

| . . 3.0 3.0 |

| . . . 1.0 |

UPLO TRANSA DIAG N AP X INCX

| | | | | | |

CALL STPMV( 'U' , 'T' , 'N' , 4 , AP , X , 1 )

AP = (1.0, 2.0, 2.0, 3.0, 2.0, 3.0, 2.0, 5.0, 3.0, 1.0)

X = (5.0, 4.0, 3.0, 2.0)

**STBMV Example 1:**

Matrix **A** is a real 7 by 7 upper triangular band matrix with a half band width of 3 that is not unit triangular, stored in upper-triangular-band-packed storage mode.

| 1.0 1.0 1.0 1.0 0.0 0.0 0.0 |

| 0.0 2.0 2.0 2.0 2.0 0.0 0.0 |

| 0.0 0.0 3.0 3.0 3.0 3.0 0.0 |

| 0.0 0.0 0.0 4.0 4.0 4.0 4.0 |

| 0.0 0.0 0.0 0.0 5.0 5.0 5.0 |

| 0.0 0.0 0.0 0.0 0.0 6.0 6.0 |

| 0.0 0.0 0.0 0.0 0.0 0.0 7.0 |

UPLO TRANSA DIAG N K A LDA X INCX

| | | | | | | | |

CALL STBMV( 'U' , 'N' , 'N' , 7 , 3 , A , 5 , X , 1 )

| . . . 1.0 2.0 3.0 4.0 |

| . . 1.0 2.0 3.0 4.0 5.0 |

A = | . 1.0 2.0 3.0 4.0 5.0 6.0 |

| 1.0 2.0 3.0 4.0 5.0 6.0 7.0 |

| . . . . . . . |

**STBMV Example 2:**

Matrix **A** is a real 7 by 7 lower triangular band matrix with a half band width of 3 that is not unit triangular, stored in lower-triangular-band-packed storage mode.

| 1.0 0.0 0.0 0.0 0.0 0.0 0.0 |

| 1.0 2.0 0.0 0.0 0.0 0.0 0.0 |

| 1.0 2.0 3.0 0.0 0.0 0.0 0.0 |

| 1.0 2.0 3.0 4.0 0.0 0.0 0.0 |

| 0.0 2.0 3.0 4.0 5.0 0.0 0.0 |

| 0.0 0.0 3.0 4.0 5.0 6.0 0.0 |

| 0.0 0.0 0.0 4.0 5.0 6.0 7.0 |

UPLO TRANSA DIAG N K A LDA X INCX

| | | | | | | | |

CALL STBMV( 'L' , 'T' , 'N' , 7 , 3 , A , 5 , X , 1 )

| 1.0 2.0 3.0 4.0 5.0 6.0 7.0 |

| 1.0 2.0 3.0 4.0 5.0 6.0 . |

A = | 1.0 2.0 3.0 4.0 5.0 . . |

| 1.0 2.0 3.0 4.0 . . . |

| . . . . . . . |

# Rank-One/Rank-Two Update

The following BLAS level 2 routines are relevant to (Update of a General Matrix):

* SGER, DGER
* CGERU, ZGERU, CGERC, ZGERC
* SSPR, DSPR, CHPR, ZHPR
* SSYR, DSYR, CHER, ZHER
* SSPR2, DSPR2, CHPR2, ZHPR2
* SSYR2, DSYR2, CHER2, ZHER2

SxxR, DxxR, CxxRU/C, and ZxxRU/C compute the rank-one update of a general matrix, using the scalar α, matrix A, vector x, and the transpose of vector y:

A ← A +αxyT

A ← A+αxyT + αyxT

The R2 indicates that there are 2 vectors; R indicates y=x.

The syntax to use these functions in different languages are summarized below:

**For general matrix:**

| **Language** | **Syntax** |
| --- | --- |
| **Fortran** | CALL SGER | DGER | CGERU | ZGERU | CGERC | ZGERC (*m*, *n*, *alpha*, *x*, *incx*, *y*, *incy*, *a*, *lda*) |
| **C and C++** | sger | dger | cgeru | zgeru | cgerc | zgerc (*m*, *n*, *alpha*, *x*, *incx*, *y*, *incy*, *a*, *lda*); |
| **CBLAS** | cblas\_sger | cblas\_dger | cblas\_cgeru | cblas\_zgeru | cblas\_cgerc | cblas\_zgerc (*cblas\_layout*, *m*, *n*, *alpha*, *x*, *incx*, *y*, *incy*, *a*, *lda*); |

**For Real Symmetric or Complex Hermitian Matrix (rank 1):**

| **Language** | **Syntax** |
| --- | --- |
| **Fortran** | CALL SSPR | DSPR | CHPR | ZHPR (*uplo*, *n*, *alpha*, *x*, *incx*, *ap*)  CALL SSYR | DSYR | CHER | ZHER (*uplo*, *n*, *alpha*, *x*, *incx*, *a*, *lda*) |
| **C and C++** | sspr | dspr | chpr | zhpr (*uplo*, *n*, *alpha*, *x*, *incx*, *ap*);  ssyr | dsyr | cher | zher (*uplo*, *n*, *alpha*, *x*, *incx*, *a*, *lda*); |
| **CBLAS** | cblas\_sspr | cblas\_dspr | cblas\_chpr | cblas\_zhpr (*cblas\_layout*, *cblas\_uplo*, *n*, *alpha*, *x*, *incx*, *ap*);  cblas\_ssyr | cblas\_dsyr | cblas\_cher | cblas\_zher (*cblas\_layout*, *cblas\_uplo*, *n*, *alpha*, *x*, *incx*, *a*, *lda*); |

**For Real Symmetric or Complex Hermitian Matrix (rank 2):**

| **Language** | **Syntax** |
| --- | --- |
| **Fortran** | CALL SSPR2 | DSPR2 | CHPR2 | ZHPR2 (*uplo*, *n*, *alpha*, *x*, *incx*, *y*, *incy*, *ap*)  CALL SSYR2 | DSYR2 | CHER2 | ZHER2 (*uplo*, *n*, *alpha*, *x*, *incx*, *y*, *incy*, *a*, *lda*) |
| **C and C++** | sspr2 | dspr2 | chpr2 | zhpr2 (*uplo*, *n*, *alpha*, *x*, *incx*, *y*, *incy*, *ap*);  ssyr2 | dsyr2 | cher2 | zher2 (*uplo*, *n*, *alpha*, *x*, *incx*, *y*, *incy*, *a*, *lda*); |
| **CBLAS** | cblas\_sspr2 | cblas\_dspr2 | cblas\_chpr2 | cblas\_zhpr2 (*cblas\_layout*, *cblas\_uplo*, *n*, *alpha*, *x*, *incx*, *y*, *incy*, *ap*);  cblas\_ssyr2 | cblas\_dsyr2 | cblas\_cher2 | cblas\_zher2 (*cblas\_layout*, *cblas\_uplo*, *n*, *alpha*, *x*, *incx*, *y*, *incy*, *a*, *lda*); |

**SGER Example:**

M N ALPHA X INCX Y INCY A LDA

| | | | | | | | |

CALL SGER( 4 , 3 , 1.0 , X , 1 , Y , 2 , A(1,0) , 10 )

X = (3.0, 2.0, 1.0, 4.0)

Y = (1.0, . , 2.0, . , 3.0)

| . . . |

| 1.0 2.0 3.0 |

| 2.0 2.0 4.0 |

| 3.0 2.0 2.0 |

| 4.0 2.0 1.0 |

A = | . . . |

| . . . |

| . . . |

| . . . |

| . . . |

Output

| . . . |

| 4.0 8.0 12.0 |

| 4.0 6.0 10.0 |

| 4.0 4.0 5.0 |

| 8.0 10.0 13.0 |

A = | . . . |

| . . . |

| . . . |

| . . . |

| . . . |

# Triangular Equations with Single Right-Hand Side or Triangular Band Equation Solve

The following BLAS level 2 routines are relevant:

* STRSV, DTRSV, CTRSV, ZTRSV, STPSV, DTPSV, CTPSV, and ZTPSV
* STBSV, DTBSV, CTBSV, and ZTBSV

SxxSV, DxxSV, CxxSV, and ZxxSV perform one of the following solves for a triangular system of equations with a single right-hand side, using the vector x and triangular matrix A or its transpose:

X ← A-1x Ax = b

x ← A-Tx ATx = b

x ← A-Hx AHx = b

The syntax to use these functions in different languages are summarized below:

**For triangular matrix:**

| **Language** | **Syntax** |
| --- | --- |
| **Fortran** | CALL STRSV | DTRSV | CTRSV | ZTRSV (*uplo*, *transa*, *diag*, *n*, *a*, *lda*, *x*, *incx*)  CALL STPSV | DTPSV | CTPSV | ZTPSV (*uplo*, *transa*, *diag*, *n*, *ap*, *x*, *incx*) |
| **C and C++** | strsv | dtrsv | ctrsv | ztrsv (*uplo*, *transa*, *diag*, *n*, *a*, *lda*, *x*, *incx*);  stpsv | dtpsv | ctpsv | ztpsv (*uplo*, *transa*, *diag*, *n*, *ap*, *x*, *incx*); |
| **CBLAS** | cblas\_strsv | cblas\_dtrsv | cblas\_ctrsv | cblas\_ztrsv (*cblas\_layout*, *cblas\_uplo*, *cblas\_transa*, *cblas\_diag*, *n*, *a*, *lda*, *x*, *incx*);  cblas\_stpsv | cblas\_dtpsv | cblas\_ctpsv | cblas\_ztpsv (*cblas\_layout*, *cblas\_uplo*, *cblas\_transa*, *cblas\_diag*, *n*, *ap*, *x*, *incx*); |

**For triangular band matrix:**

| **Language** | **Syntax** |
| --- | --- |
| **Fortran** | CALL STBSV | DTBSV | CTBSV | ZTBSV (*uplo*, *trans*, *diag*, *n*, *k*, *a*, *lda*, *x*, *incx*) |
| **C and C++** | stbsv | dtbsv | ctbsv | ztbsv (*uplo*, *trans*, *diag*, *n*, *k*, *a*, *lda*, *x*, *incx*);f |
| **CBLAS** | cblas\_stbsv | cblas\_dtbsv | cblas\_ctbsv | cblas\_ztbsv (*cblas\_layout*, *cblas\_uplo*, *cblas\_trans*, *cblas\_diag*, *n*, *k*, *a*, *lda*, *x*, *incx*); |

**STRSV Example:**

UPLO TRANSA DIAG N A LDA X INCX

| | | | | | | |

CALL STRSV( 'L' , 'N' , 'U' , 4 , A , 4 , X , 1 )

| . . . . |

| 1.0 . . . |

A = | 2.0 3.0 . . |

| 3.0 4.0 3.0 . |

X = (1.0, 3.0, 11.0, 24.0)

Output:

X = (1.0, 2.0, 3.0, 4.0)

**STRSV Example:**

UPLO TRANSA DIAG N A LDA X INCX

| | | | | | | |

CALL STRSV( 'U' , 'T' , 'N' , 4 , A , 4 , X , 1 )

| 1.0 2.0 3.0 2.0 |

A = | . 2.0 2.0 5.0 |

| . . 3.0 3.0 |

| . . . 1.0 |

X = (5.0, 18.0, 32.0, 41.0)

Output:

X = (5.0, 4.0, 3.0, 2.0)