NISTIR 6744

NIST Sparse BLAS User's Guide

Roldan Pozo
Karin A. Remington

U. S. DEPARTMENT OF COMMERCE
Technology Administration
National Institute of Standards
and Technology
Gaithersburg, MD 20899-8230

1901-2001
NIST CENTENNIAL

NIST
National Institute of Standards and Technology
Technology Administration
U.S. Department of Commerce

NISTIR 6744

NIST Sparse BLAS User's Guide

Roldan Pozo
Karin A. Remington

U. S. DEPARTMENT OF COMMERCE
Technology Administration
National Institute of Standards
and Technology
Gaithersburg, MD 20899-8230

May 9, 2001

U.S. DEPARTMENT OF COMMERCE
Donald L. Evans, Secretary

NATIONAL INSTITUTE OF STANDARDS
AND TECHNOLOGY
Dr. Karen H. Brown, Acting Director

NIST Sparse BLAS
User's Guide

Karin A. Remington* and Roldan Pozo*
National Institute of Standards and Technology

July 1996

*e-mail: [kremington,pozo]@nist.gov

1 Introduction

The NIST Sparse BLAS (Basic Linear Algebra Subprogram) library provides computational kernels for fundamental sparse matrix operations:

- sparse matrix products,

$$C \leftarrow \alpha\, A\, B + \beta C$$
$$C \leftarrow \alpha\, A^T\, B + \beta C$$

- solution of triangular systems,

$$C \leftarrow \alpha D_L\, A^{-1}\, D_R B + \beta C$$
$$C \leftarrow \alpha D_L\, A^{-T}\, D_R B + \beta C$$

where A is sparse matrix, B and C are dense matrices/vectors, and D_L and D_R are diagonal matrices. This version of the NIST Sparse BLAS supports the following sparse formats: compressed sparse row (CSR), compressed sparse column (CSC), coordinate (COO), block sparse row (BSR), block sparse column (BSC), block coordinate (BCO) and variable block row (VBR). Symmetric and skew-symmetric versions are also supported.

The routines are written in ANSI C and are callable from Fortran and C through the interface proposed in the **Sparse BLAS Toolkit**[1]. Also see the companion paper [2].

In addition to the Sparse BLAS Toolkit interface, developers have access to lightweight kernel routines. These **Sparse BLAS Lite** routines are unique to each parameter combination of the higher-level Toolkit interface. The Lite routines are designed for minimal overhead; they have no case statements, nor elaborate error-detection overhead. Thus, they are ideal for use on small matrices or to be used as efficient building blocks in higher-level routines. Some typical examples of the Lite routines:

```
C <- A' * B              CSR_MatMult_CATB_double()
C <- A * B + C           CSR_MatMult_CABC_double()
C <- alpha*A*B + b*C     CSR_MatMult_CaBbC_double()
C <- D*A^(-1)*B + C      CSR_MatTriangSlvLD_CDABC_double()
```

These lightweight kernel routines are generated from a small number of source lines (less than 5000 for the storage formats currently supported) by defining and expanding macros for successively restrictive sets of calling sequence parameters. This allows changes to the core source code, made for optimization or debugging, to be rapidly and automatically propagated to all affected kernel routines (approximately 130,000 lines of code).

Section 2 gives an introduction to the source code generation mechanism for Toolkit's underlying "Lite" kernel library. Section 3 provides interface specifications for the Toolkit routines provided in this release, and Section 4 gives the function prototypes for the "Lite" interface routines for the VBR (variable block row) format as an example. Prototypes for other formats are similar, and can be obtained directly from the header files in the include subdirectory. Installation instructions for the library are provided in Appendix A.

Note for Fortran Users: The interfaces described in this user guide are C interfaces. For the Toolkit layer, the Fortran interfaces are similar, except that all arguments are passed by *reference* (that is, typical Fortran style). The "Lite" interface is not currently available in Fortran, because of the need for long routine names. This restriction will be re-examined for future releases.

2 Source Code Generation

The SRC_GEN directory contains the following generic source code files.

`bcomm.c`	`bsrmm.c`	`cscmm.c`	`csrmts.c`
`bscmm.c`	`bsrmts.c`	`cscmts.c`	`vbrmm.c`
`bscmts.c`	`coomm.c`	`csrmm.c`	`vbrmts.c`

Also provided are generator scripts for creating the NIST Sparse BLAS kernel routines from these generic source files.

These source files are used as "master files", and are written in such a way that special case routines can be generated by relatively simple shell scripts which use "sed" and "awk" for text replacement. The approach saves considerable programming effort by generating most source files automatically, and reduces errors by insuring that any changes are propagated throughout all of the related source code.

The master files provide working source code for the most general version of the kernel routine. This is where real programming effort should be expended to optimized the library. The code is commented with tags which can be used to selectively delete code for special case routines. The "rules" for creating each special case file are defined in the SRC_GEN/kernels subdirectory. The kernels subdirectory contains the files

`CAB`	`CADBbC`	`CDADBC`	`CaADB`	`CaDABbC`
`CABC`	`CDAB`	`CDADBbC`	`CaADBC`	`CaDADB`
`CABbC`	`CDABC`	`CaAB`	`CaADBbC`	`CaDADBC`
`CADB`	`CDABbC`	`CaABC`	`CaDAB`	`CaDADBbC`
`CADBC`	`CDADB`	`CaABbC`	`CaDABC`	

one representing each of the specializations from the generic master code, along with kernel files for the master codes. Each of these kernel files contains pointers to appropriate "Definition" files, in the directory SRC_GEN/Defs, which are used to build up the sed script for the text replacement to generate the kernel routines.

For typical use, these kernel and definition files would never have to be touched. Many modifications (say for optimization) can be made to the master source files without requiring any change whatsoever to the file generation mechanism. The only source code changes which would affect code generation would be those which alter the relationship between the comment tags and the related source. A more detailed explanation of the mechanism, and requirements for modifications, will be forthcoming in the 1.0 release.

After making any necessary changes to these "master" source files, the library source files may be generated via the "create" script (automated in the "make" process in this directory with "make install" or "make re-install").

**** IMPORTANT NOTE ****

Any changes to source for any routines below the Toolkit interface layer **MUST** be made in the ../SRC_GEN directory to be retained and propagated to all appropriate kernel routines. Changes to the Toolkit interface routines, however, should be made directly in the directory `../src_tk[c|f]`.)

3 Toolkit Interface Descriptions

Name

dbcomm

Calling Sequence

```
void  dbcomm( const int transa, const int mb, const int n, const int kb,
              const double alpha, const int descra[], const double val[],
              const int bindx[], const int bjndx[], const int bnnz,
              const int lb, const double b[], const int ldb,
              const double beta, double c[], const int ldc,
              double work[], const int lwork);
```

Functionality

Block coordinate format matrix-matrix multiply.

$$C \leftarrow \alpha AB + \beta C$$

$$C \leftarrow \alpha A^T B + \beta C$$

Arguments

int transa	Indicates how to operate with the sparse matrix 0 : operate with matrix 1 : operate with transpose matrix
int mb	Number of block rows in matrix A
int n	Number of columns in matrix c
int kb	Number of block columns in matrix A
double alpha	Scalar parameter
double beta	Scalar parameter
int descra[]	Descriptor argument. Five element integer array: descra[0] matrix structure 0 : general 1 : symmetric 2 : Hermitian 3 : Triangular 4 : Skew(Anti)-Symmetric 5 : Diagonal descra[1] upper/lower triangular indicator 1 : lower 2 : upper descra[2] main diagonal type 0 : non-unit 1 : unit descra[3] Array base 0 : C/C++ compatible 1 : Fortran compatible descra[4] repeated indices (not currently supported) 0 : unknown

	1 : no repeated indices
double *val	scalar array of length nnz containing matrix entries
int *bindx	integer array of length bnnz consisting of the block row indices of the entries of A.
int *bjndx	integer array of length bnnz consisting of the block column indices of the entries of A.
int bnnz	number of block entries
int lb	dimension of blocks
double *b	rectangular array with leading dimension ldb
int ldb	leading dimension of b
double *c	rectangular array with leading dimension ldc
int ldc	leading dimension of c
double *work	scratch array of length lwork. lwork should be at least max(m,n)
int lwork	length of work array

Name

dbscmm

Calling Sequence

```
void  dbscmm( const int transa, const int mb, const int n, const int kb,
              const double alpha, const int descra[], const double val[],
              const int bindx[], const int bpntrb[], const int bpntre[],
              const int lb, const double b[], const int ldb,
              const double beta, double c[], const int ldc,
              double work[], const int lwork);
```

Functionality

Block sparse column format matrix-matrix multiply.

$$C \leftarrow \alpha AB + \beta C$$

$$C \leftarrow \alpha A^T B + \beta C$$

Arguments

int transa	Indicates how to operate with the sparse matrix 0 : operate with matrix 1 : operate with transpose matrix
int mb	Number of block rows in matrix A
int n	Number of columns in matrix c
int kb	Number of block columns in matrix A
double alpha	Scalar parameter
double beta	Scalar parameter
int descra[]	Descriptor argument. Five element integer array: descra[0] matrix structure 0 : general 1 : symmetric 2 : Hermitian 3 : Triangular 4 : Skew(Anti)-Symmetric 5 : Diagonal descra[1] upper/lower triangular indicator 1 : lower 2 : upper descra[2] main diagonal type 0 : non-unit 1 : unit descra[3] Array base 0 : C/C++ compatible 1 : Fortran compatible descra[4] repeated indices (not currently supported) 0 : unknown

	1 : no repeated indices
double *val	scalar array of length nnz containing matrix entries
int *bindx	integer array of length bnnz consisting of the block row indices of the entries of A.
int *bpntrb	integer array of length mb such that bpntrb(i)-bpntrb(1) points to location in bindx of the first block entry of the j-th column of A.
int *bpntre	integer array of length mb such that bpntre(i)-bpntrb(1) points to location in bindx of the last block entry of the j-th column of A.
int lb	dimension of blocks
double *b	rectangular array with leading dimension ldb
int ldb	leading dimension of b
double *c	rectangular array with leading dimension ldc
int ldc	leading dimension of c
double *work	scratch array of length lwork. lwork should be at least max(m,n)
int lwork	length of work array

Name

dbscsm

Calling Sequence

```
void  dbscsm( const int transa, const int mb, const int n,
              const int unitd, const double dv[],
              const double alpha, const int descra[], const double val[],
              const int bindx[], const int bpntrb[], const int bpntre[],
              const int lb, const double b[], const int ldb,
              const double beta, double c[], const int ldc,
              double work[], const int lwork);
```

Functionality

Block sparse column format triangular solve.

$$C \leftarrow \alpha D A^{-1} B + \beta C \qquad C \leftarrow \alpha D A^{-T} B + \beta C$$

$$C \leftarrow \alpha A^{-1} D B + \beta C \qquad C \leftarrow \alpha A^{-T} D B + \beta C$$

Arguments

int transa	Indicates how to operate with the sparse matrix 0 : operate with matrix 1 : operate with transpose matrix
int mb	Number of block rows in matrix A
int n	Number of columns in matrix c
int unitd	Type of scaling: 1 : Identity matrix (argument dv[] is ignored) 2 : Scale on left (row scaling) 3 : Scale on right (column scaling)
double alpha	Scalar parameter
double beta	Scalar parameter
int descra[]	Descriptor argument. Five element integer array: descra[0] matrix structure 0 : general 1 : symmetric 2 : Hermitian 3 : Triangular 4 : Skew(Anti)-Symmetric 5 : Diagonal descra[1] upper/lower triangular indicator 1 : lower 2 : upper descra[2] main diagonal type 0 : non-unit 1 : unit descra[3] Array base

	0 : C/C++ compatible
	1 : Fortran compatible
	descra[4] repeated indices (not currently supported)
	0 : unknown
	1 : no repeated indices
double *val	scalar array of length nnz containing matrix entries
int *bindx	integer array of length bnnz consisting of the block row indices of the entries of A.
int *bpntrb	integer array of length mb such that bpntrb(i)-bpntrb(1) points to location in bindx of the first block entry of the j-th column of A.
int *bpntre	integer array of length mb such that bpntre(i)-bpntrb(1) points to location in bindx of the last block entry of the j-th column of A.
int bnnz	number of block entries
int lb	dimension of blocks
double *b	rectangular array with leading dimension ldb
int ldb	leading dimension of b
double *c	rectangular array with leading dimension ldc
int ldc	leading dimension of c
double *work	scratch array of length lwork. lwork should be at least max(m,n)
int lwork	length of work array

Name

dbsrmm

Calling Sequence

```
void  dbsrmm( const int transa, const int mb, const int n, const int kb,
              const double alpha, const int descra[], const double val[],
              const int bindx[], const int bpntrb[], const int bpntre[],
              const int lb, const double b[], const int ldb,
              const double beta, double c[], const int ldc,
              double work[], const int lwork);
```

Functionality

Block sparse row format matrix-matrix multiply.

$$C \leftarrow \alpha AB + \beta C$$

$$C \leftarrow \alpha A^T B + \beta C$$

Arguments

int transa	Indicates how to operate with the sparse matrix 0 : operate with matrix 1 : operate with transpose matrix
int mb	Number of block rows in matrix A
int n	Number of columns in matrix c
int kb	Number of block columns in matrix A
double alpha	Scalar parameter
double beta	Scalar parameter
int descra[]	Descriptor argument. Five element integer array: descra[0] matrix structure 0 : general 1 : symmetric 2 : Hermitian 3 : Triangular 4 : Skew(Anti)-Symmetric 5 : Diagonal descra[1] upper/lower triangular indicator 1 : lower 2 : upper descra[2] main diagonal type 0 : non-unit 1 : unit descra[3] Array base 0 : C/C++ compatible 1 : Fortran compatible descra[4] repeated indices (not currently supported) 0 : unknown

	1 : no repeated indices
double *val	scalar array of length nnz containing matrix entries
int *bindx	integer array of length bnnz consisting of the block column indices of the entries of A.
int *bpntrb	integer array of length mb such that bpntrb(i)-bpntrb(1) points to location in bindx of the first block entry of the j-th row of A.
int *bpntre	integer array of length mb such that bpntre(i)-bpntrb(1) points to location in bindx of the last block entry of the j-th row of A.
int lb	dimension of blocks
double *b	rectangular array with leading dimension ldb
int ldb	leading dimension of b
double *c	rectangular array with leading dimension ldc
int ldc	leading dimension of c
double *work	scratch array of length lwork. lwork should be at least max(m,n)
int lwork	length of work array

Name

dbsrsm

Calling Sequence

```
void  dbsrsm( const int transa, const int mb, const int n,
              const int unitd, const double dv[],
              const double alpha, const int descra[], const double val[],
              const int bindx[], const int bpntrb[], const int bpntre[],
              const int lb, const double b[], const int ldb,
              const double beta, double c[], const int ldc,
              double work[], const int lwork);
```

Functionality

Block sparse row format triangular solve.

$$C \leftarrow \alpha D A^{-1} B + \beta C \qquad C \leftarrow \alpha D A^{-T} B + \beta C$$

$$C \leftarrow \alpha A^{-1} D B + \beta C \qquad C \leftarrow \alpha A^{-T} D B + \beta C$$

Arguments

int transa	Indicates how to operate with the sparse matrix 0 : operate with matrix 1 : operate with transpose matrix
int mb	Number of block rows in matrix A
int n	Number of columns in matrix c
int unitd	Type of scaling: 1 : Identity matrix (argument dv[] is ignored) 2 : Scale on left (row scaling) 3 : Scale on right (column scaling)
double alpha	Scalar parameter
double beta	Scalar parameter
int descra[]	Descriptor argument. Five element integer array: descra[0] matrix structure 0 : general 1 : symmetric 2 : Hermitian 3 : Triangular 4 : Skew(Anti)-Symmetric 5 : Diagonal descra[1] upper/lower triangular indicator 1 : lower 2 : upper descra[2] main diagonal type 0 : non-unit 1 : unit descra[3] Array base

	0 : C/C++ compatible
	1 : Fortran compatible
	descra[4] repeated indices (not currently supported)
	0 : unknown
	1 : no repeated indices
double *val	scalar array of length nnz containing matrix entries
int *bindx	integer array of length bnnz consisting of the block column indices of the entries of A.
int *bpntrb	integer array of length mb such that bpntrb(i)-bpntrb(1) points to location in bindx of the first block entry of the j-th row of A.
int *bpntre	integer array of length mb such that bpntre(i)-bpntrb(1) points to location in bindx of the last block entry of the j-th row of A.
int lb	dimension of blocks
double *b	rectangular array with leading dimension ldb
int ldb	leading dimension of b
double *c	rectangular array with leading dimension ldc
int ldc	leading dimension of c
double *work	scratch array of length lwork. lwork should be at least max(m,n)
int lwork	length of work array

Name

dcoomm

Calling Sequence

```
void  dcoomm( const int transa, const int m, const int n, const int k,
              const double alpha, const int descra[], const double val[],
              const int indx[], const int jndx[], const int nnz,
              const double b[], const int ldb,
              const double beta, double c[], const int ldc,
              double work[], const int lwork);
```

Functionality

Coordinate format matrix-matrix multiply.

$$C \leftarrow \alpha AB + \beta C$$

$$C \leftarrow \alpha A^T B + \beta C$$

Arguments

int transa	Indicates how to operate with the sparse matrix 0 : operate with matrix 1 : operate with transpose matrix
int m	Number of rows in matrix A
int n	Number of columns in matrix c
int k	Number of columns in matrix A
double alpha	Scalar parameter
double beta	Scalar parameter
int descra[]	Descriptor argument. Five element integer array: descra[0] matrix structure 0 : general 1 : symmetric 2 : Hermitian 3 : Triangular 4 : Skew(Anti)-Symmetric 5 : Diagonal descra[1] upper/lower triangular indicator 1 : lower 2 : upper descra[2] main diagonal type 0 : non-unit 1 : unit descra[3] Array base 0 : C/C++ compatible 1 : Fortran compatible descra[4] repeated indices (not currently supported) 0 : unknown

	1 : no repeated indices
double *val	scalar array of length nnz containing matrix entries
int *indx	integer array of length nnz containing row indices
int *jndx	integer array of length nnz containing column indices
int nnz	Number of nonzero matrix entries
double *b	rectangular array with leading dimension ldb
int ldb	leading dimension of b
double *c	rectangular array with leading dimension ldc
int ldc	leading dimension of c
double *work	scratch array of length lwork. lwork should be at least max(m,n)

Name

dcscmm

Calling Sequence

```
void  dcscmm( const int transa, const int m, const int n, const int k,
              const double alpha, const int descra[], const double val[],
              const int indx[], const int pntrb[], const int pntre[],
              const double b[], const int ldb,
              const double beta, double c[], const int ldc,
              double work[], const int lwork);
```

Functionality

Compressed sparse column format matrix-matrix multiply.

$$C \leftarrow \alpha AB + \beta C$$

$$C \leftarrow \alpha A^T B + \beta C$$

Arguments

int transa	Indicates how to operate with the sparse matrix 0 : operate with matrix 1 : operate with transpose matrix
int m	Number of rows in matrix A
int n	Number of columns in matrix c
int k	Number of columns in matrix A
double alpha	Scalar parameter
double beta	Scalar parameter
int descra[]	Descriptor argument. Five element integer array: descra[0] matrix structure 0 : general 1 : symmetric 2 : Hermitian 3 : Triangular 4 : Skew(Anti)-Symmetric 5 : Diagonal descra[1] upper/lower triangular indicator 1 : lower 2 : upper descra[2] main diagonal type 0 : non-unit 1 : unit descra[3] Array base 0 : C/C++ compatible 1 : Fortran compatible descra[4] repeated indices (not currently supported) 0 : unknown

	1 : no repeated indices
double *val	scalar array of length nnz containing matrix entries
int *indx	integer array of length nnz containing row indices
int *pntrb	integer array of length k such that pntrb(j)-pntrb(1) points to location in val of the first nonzero element in column j
int *pntre	integer array of length k such that pntre(j)-pntrb(1) points to location in val of the last nonzero element in column j
double *b	rectangular array with leading dimension ldb
int ldb	leading dimension of b
double *c	rectangular array with leading dimension ldc
int ldc	leading dimension of c
double *work	scratch array of length lwork. lwork should be at least max(m,n)

Name

dcscsm

Calling Sequence

```
void  dcscsm( const int transa, const int m, const int n,
              const int unitd, const double dv[],
              const double alpha, const int descra[], const double val[],
              const int indx[], const int pntrb[], const int pntre[],
              const double b[], const int ldb,
              const double beta, double c[], const int ldc,
              double work[], const int lwork);
```

Functionality

Compressed sparse column format triangular solve.

$$C \leftarrow \alpha D A^{-1} B + \beta C \qquad C \leftarrow \alpha D A^{-T} B + \beta C$$

$$C \leftarrow \alpha A^{-1} D B + \beta C \qquad C \leftarrow \alpha A^{-T} D B + \beta C$$

Arguments

int transa — Indicates how to operate with the sparse matrix
0 : operate with matrix
1 : operate with transpose matrix

int m — Number of rows in matrix A

int n — Number of columns in matrix c

int unitd — Type of scaling:
- 1 : Identity matrix (argument dv[] is ignored)
- 2 : Scale on left (row scaling)
- 3 : Scale on right (column scaling)

double alpha — Scalar parameter

double beta — Scalar parameter

int descra[] — Descriptor argument. Five element integer array:

descra[0] matrix structure
- 0 : general
- 1 : symmetric
- 2 : Hermitian
- 3 : Triangular
- 4 : Skew(Anti)-Symmetric
- 5 : Diagonal

descra[1] upper/lower triangular indicator
- 1 : lower
- 2 : upper

descra[2] main diagonal type
- 0 : non-unit
- 1 : unit

descra[3] Array base

	0 : C/C++ compatible
	1 : Fortran compatible
	descra[4] repeated indices (not currently supported)
	0 : unknown
	1 : no repeated indices
double *val	scalar array of length nnz containing matrix entries
int *indx	integer array of length nnz containing row indices
int *pntrb	integer array of length k such that pntrb(j)-pntrb(1) points to location in val of the first nonzero element in column j
int *pntre	integer array of length k such that pntre(j)-pntrb(1) points to location in val of the last nonzero element in column j
double *b	rectangular array with leading dimension ldb
int ldb	leading dimension of b
double *c	rectangular array with leading dimension ldc
int ldc	leading dimension of c
double *work	scratch array of length lwork. lwork should be at least max(m,n)
int lwork	length of work array

Name

dcsrmm

Calling Sequence

```
void  dcsrmm( const int transa, const int m, const int n, const int k,
              const double alpha, const int descra[], const double val[],
              const int indx[], const int pntrb[], const int pntre[],
              const double b[], const int ldb,
              const double beta, double c[], const int ldc,
              double work[], const int lwork);
```

Functionality

Compressed sparse row format matrix-matrix multiply.

$$C \leftarrow \alpha AB + \beta C$$

$$C \leftarrow \alpha A^T B + \beta C$$

Arguments

int transa	Indicates how to operate with the sparse matrix 0 : operate with matrix 1 : operate with transpose matrix
int m	Number of rows in matrix A
int n	Number of columns in matrix c
int k	Number of columns in matrix A
double alpha	Scalar parameter
double beta	Scalar parameter
int descra[]	Descriptor argument. Five element integer array: descra[0] matrix structure 0 : general 1 : symmetric 2 : Hermitian 3 : Triangular 4 : Skew(Anti)-Symmetric 5 : Diagonal descra[1] upper/lower triangular indicator 1 : lower 2 : upper descra[2] main diagonal type 0 : non-unit 1 : unit descra[3] Array base 0 : C/C++ compatible 1 : Fortran compatible descra[4] repeated indices (not currently supported) 0 : unknown

	1 : no repeated indices
double *val	scalar array of length nnz containing matrix entries
int *indx	integer array of length nnz containing column indices
int *pntrb	integer array of length k such that pntrb(j)-pntrb(1) points to location in val of the first nonzero element in row j
int *pntre	integer array of length k such that pntre(j)-pntrb(1) points to location in val of the last nonzero element in row j
double *b	rectangular array with leading dimension ldb
int ldb	leading dimension of b
double *c	rectangular array with leading dimension ldc
int ldc	leading dimension of c
double *work	scratch array of length lwork. lwork should be at least max(m,n)

Name

dcsrsm

Calling Sequence

```
void  dcsrsm( const int transa, const int m, const int n,
              const int unitd, const double dv[],
              const double alpha, const int descra[], const double val[],
              const int indx[], const int pntrb[], const int pntre[],
              const double b[], const int ldb,
              const double beta, double c[], const int ldc,
              double work[], const int lwork);
```

Functionality

Compressed sparse row format triangular solve.

$$C \leftarrow \alpha D A^{-1} B + \beta C \qquad C \leftarrow \alpha D A^{-T} B + \beta C$$

$$C \leftarrow \alpha A^{-1} D B + \beta C \qquad C \leftarrow \alpha A^{-T} D B + \beta C$$

Arguments

int transa — Indicates how to operate with the sparse matrix
- 0 : operate with matrix
- 1 : operate with transpose matrix

int m — Number of rows in matrix A

int n — Number of columns in matrix c

int unitd — Type of scaling:
- 1 : Identity matrix (argument dv[] is ignored)
- 2 : Scale on left (row scaling)
- 3 : Scale on right (column scaling)

double alpha — Scalar parameter

double beta — Scalar parameter

int descra[] — Descriptor argument. Five element integer array:

descra[0] matrix structure
- 0 : general
- 1 : symmetric
- 2 : Hermitian
- 3 : Triangular
- 4 : Skew(Anti)-Symmetric
- 5 : Diagonal

descra[1] upper/lower triangular indicator
- 1 : lower
- 2 : upper

descra[2] main diagonal type
- 0 : non-unit
- 1 : unit

descra[3] Array base

	0 : C/C++ compatible
	1 : Fortran compatible
	descra[4] repeated indices (not currently supported)
	0 : unknown
	1 : no repeated indices
double *val	scalar array of length nnz containing matrix entries
int *indx	integer array of length nnz containing column indices
int *pntrb	integer array of length k such that pntrb(j)-pntrb(1) points to location in val of the first nonzero element in row j
int *pntre	integer array of length k such that pntre(j)-pntrb(1) points to location in val of the last nonzero element in row j
double *b	rectangular array with leading dimension ldb
int ldb	leading dimension of b
double *c	rectangular array with leading dimension ldc
int ldc	leading dimension of c
double *work	scratch array of length lwork. lwork should be at least max(m,n)
int lwork	length of work array

Name

dvbrmm

Calling Sequence

```
void  dvbrmm( const int transa, const int mb, const int n, const int kb,
              const double alpha, const int descra[], const double val[],
              const int indx[], const int bindx[],
              const int rpntr[], const int cpntr[],
              const int bpntrb[], const int bpntre[],
              const double b[], const int ldb,
              const double beta, double c[], const int ldc,
              double work[], const int lwork);
```

Functionality

Variable block row format matrix-matrix multiply.

$$C \leftarrow \alpha AB + \beta C$$

$$C \leftarrow \alpha A^T B + \beta C$$

Arguments

int transa	Indicates how to operate with the sparse matrix 0 : operate with matrix 1 : operate with transpose matrix
int mb	Number of block rows in matrix A
int n	Number of columns in matrix c
int kb	Number of block columns in matrix A
double alpha	Scalar parameter
double beta	Scalar parameter
int descra[]	Descriptor argument. Five element integer array: descra[0] matrix structure 0 : general 1 : symmetric 2 : Hermitian 3 : Triangular 4 : Skew(Anti)-Symmetric 5 : Diagonal descra[1] upper/lower triangular indicator 1 : lower 2 : upper descra[2] main diagonal type 0 : non-unit 1 : unit descra[3] Array base 0 : C/C++ compatible 1 : Fortran compatible

	descra[4] repeated indice (not currently supported)
s	0 : unknown
	1 : no repeated indices
double *val	scalar array of length nnz containing matrix entries
int *indx	integer array of length bnnz+1 such that the i-th element of indx[] points to the location in val of the (1,1) element of the i-th block entry.
int *bindx	integer array of length bnnz consisting of the block column indices of the entries of A.
int *rpntr	integer array of length mb+1 such that rpntr(i)-rpntr(1) is the row index of the first point row in the i-th block row. rpntr(mb+1) is set to m+rpntr(1). Thus, the number of point rows in the i-th block row is rpntr(i+1)-rpntr(i).
int *cpntr	integer array of length kb+1 such that cpntr(j)-cpntr(1) is the column index of the first point column in the j-th block column. cpntr(kb+1) is set to k+cpntr(1). Thus, the number of point columns in the j-th block column is cpntr(j+1)-cpntr(j).
int *bpntrb	integer array of length mb such that bpntrb(i)-bpntrb(1) points to location in bindx of the first block entry of the j-th row of A.
int *bpntre	integer array of length mb such that bpntre(i)-bpntrb(1) points to location in bindx of the last block entry of the j-th row of A.
double *b	rectangular array with leading dimension ldb
int ldb	leading dimension of b
double *c	rectangular array with leading dimension ldc
int ldc	leading dimension of c
double *work	scratch array of length lwork. lwork should be at least m*n + max(blocksize)$\hat{2}$
int lwork	length of work array

Name

dvbrsm

Calling Sequence

```
void  dvbrsm( const int transa, const int mb, const int n,
              const int unitd, const double dv[],
              const double alpha, const int descra[], const double val[],
              const int indx[], const int bindx[], const int rpntr[],
              const int cpntr[], const int bpntrb[], const int bpntre[],
              const double b[], const int ldb,
              const double beta, double c[], const int ldc,
              double work[], const int lwork);
```

Functionality

Variable block row format triangular solve.

$$C \leftarrow \alpha D A^{-1} B + \beta C \qquad C \leftarrow \alpha D A^{-T} B + \beta C$$

$$C \leftarrow \alpha A^{-1} D B + \beta C \qquad C \leftarrow \alpha A^{-T} D B + \beta C$$

Arguments

int transa	Indicates how to operate with the sparse matrix 0 : operate with matrix 1 : operate with transpose matrix
int mb	Number of block rows in matrix A
int n	Number of columns in matrix c
int unitd	Type of scaling: 1 : Identity matrix (argument dv[] is ignored) 2 : Scale on left (row scaling) 3 : Scale on right (column scaling)
double alpha	Scalar parameter
double beta	Scalar parameter
int descra[]	Descriptor argument. Five element integer array: descra[0] matrix structure 0 : general 1 : symmetric 2 : Hermitian 3 : Triangular 4 : Skew(Anti)-Symmetric 5 : Diagonal descra[1] upper/lower triangular indicator 1 : lower 2 : upper descra[2] main diagonal type 0 : non-unit 1 : unit

	descra[3] Array base
	0 : C/C++ compatible
	1 : Fortran compatible
	descra[4] repeated indices (not currently supported)
	0 : unknown
	1 : no repeated indices
double *val	scalar array of length nnz containing matrix entries
int *indx	integer array of length bnnz+1 such that the i-th element of indx[] points to the location in val of the (1,1) element of the i-th block entry.
int *bindx	integer array of length bnnz consisting of the block column indices of the entries of A.
int *rpntr	integer array of length mb+1 such that rpntr(i)-rpntr(1) is the row index of the first point row in the i-th block row. rpntr(mb+1) is set to m+rpntr(1). Thus, the number of point rows in the i-th block row is rpntr(i+1)-rpntr(i).
int *cpntr	integer array of length kb+1 such that cpntr(j)-cpntr(1) is the column index of the first point column in the j-th block column. cpntr(kb+1) is set to k+cpntr(1). Thus, the number of point columns in the j-th block column is cpntr(j+1)-cpntr(j).
int *bpntrb	integer array of length mb such that bpntrb(i)-bpntrb(1) points to location in bindx of the first block entry of the j-th row of A.
int *bpntre	integer array of length mb such that bpntre(i)-bpntrb(1) points to location in bindx of the last block entry of the j-th row of A.
double *b	rectangular array with leading dimension ldb
int ldb	leading dimension of b
double *c	rectangular array with leading dimension ldc
int ldc	leading dimension of c
double *work	scratch array of length lwork. lwork should be at least m*n + max(blocksize)$\hat{2}$
int lwork	length of work array

4 "Lite" Function Prototypes: VBR Example

This section is provided to give an illustration of the naming convention and corresponding prototypes for the automatically generating lightweight functions. Since there are over 250 functions for the VBR storage format alone, it is important that these functions and prototypes follow a predictable scheme. If a user is familiar with the prototype for the most generic function (corresponding to the matrix matrix multiplication kernel CaABbC and the matrix triangular solve kernel CaDADBbC), then special case prototypes can be predicted by changing the kernel name and dropping out any corresponding unnecessary arguments from the argument list. For example, the vector version of the routine

void VBR_MatTriangSlvLU_CaDADBbC_double(`const int mb, const int n, const double *dvl, const double *dvr, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, const double beta, double *c, const int ldc, double *work, const int ind_base);`

can be obtained by changing the function name to **VBR_<u>Vec</u>TriangSlvLU_CaDADBbC_double**, and eliminating the arguments { `n, ldb, ldc`, to arrive at the prototype:

void VBR_VecTriangSlvLU_CaDADBbC_double(`const int mb, const double *dvl, const double *dvr, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const double beta, double *c, double *work, const int ind_base);`

The protoypes listed in this section further illustrate the use of this convention.

4.1 Variable Block Row Matrix Multiply Routines

void VBR_MatMult_CAB_double(const int mb, const int n, const int kb, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, double *c, const int ldc, const int ind_base);

void VBR_MatMult_CATB_double(const int mb, const int n, const int kb, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, double *c, const int ldc, const int ind_base);

void VBRsymm_MatMult_CAB_double(const int mb, const int n, const int kb, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, double *c, const int ldc, const int ind_base);

void VBRskew_MatMult_CAB_double(const int mb, const int n, const int kb, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, double *c, const int ldc, const int ind_base);

void VBRskew_MatMult_CATB_double(const int mb, const int n, const int kb, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, double *c, const int ldc, const int ind_base);

void VBR_MatMult_CaAB_double(const int mb, const int n, const int kb, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, double *c, const int ldc, const int ind_base);

void VBR_MatMult_CaATB_double(const int mb, const int n, const int kb, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, double *c, const int ldc, const int ind_base);

void VBRsymm_MatMult_CaAB_double(const int mb, const int n, const int kb, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, double *c, const int ldc, const int ind_base);

void VBRskew_MatMult_CaAB_double(const int mb, const int n, const int kb, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, double *c, const int ldc, const int ind_base);

void VBRskew_MatMult_CaATB_double(const int mb, const int n, const int kb, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, double *c, const int ldc, const int ind_base);

void VBR_MatMult_CABC_double(const int mb, const int n, const int kb, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, double *c, const int ldc, const int ind_base);

void VBR_MatMult_CATBC_double(const int mb, const int n, const int kb, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, double *c, const int ldc, const int ind_base);

void VBRsymm_MatMult_CABC_double(const int mb, const int n, const int kb, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, double *c, const int ldc, const int ind_base);

void VBRskew_MatMult_CABC_double(const int mb, const int n, const int kb, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, double *c, const int ldc, const int ind_base);

void VBRskew_MatMult_CATBC_double(const int mb, const int n, const int kb, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, double *c, const int ldc, const int ind_base);

void VBR_MatMult_CaABC_double(const int mb, const int n, const int kb, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, double *c, const int ldc, const int ind_base);

void VBR_MatMult_CaATBC_double(const int mb, const int n, const int kb, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, double *c, const int ldc, const int ind_base);

void VBRsymm_MatMult_CaABC_double(const int mb, const int n, const int kb, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, double *c, const int ldc, const int ind_base);

void VBRskew_MatMult_CaABC_double(const int mb, const int n, const int kb, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, double *c, const int ldc, const int ind_base);

void VBRskew_MatMult_CaATBC_double(const int mb, const int n, const int kb, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, double *c, const int ldc, const int ind_base);

void VBR_MatMult_CABbC_double(const int mb, const int n, const int kb, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, const double beta, double *c, const int ldc, const int ind_base);

void VBR_MatMult_CATBbC_double(const int mb, const int n, const int kb, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, const double beta, double *c, const int ldc, const int ind_base);

void VBRsymm_MatMult_CABbC_double(const int mb, const int n, const int kb, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, const double beta, double *c, const int ldc, const int ind_base);

void VBRskew_MatMult_CABbC_double(const int mb, const int n, const int kb, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, const double beta, double *c, const int ldc, const int ind_base);

void VBRskew_MatMult_CATBbC_double(const int mb, const int n, const int kb, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, const double beta, double *c, const int ldc, const int ind_base);

void VBR_MatMult_CaABbC_double(const int mb, const int n, const int kb, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, const double beta, double *c, const int ldc, const int ind_base);

void VBR_MatMult_CaATBbC_double(const int mb, const int n, const int kb, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, const double beta, double *c, const int ldc, const int ind_base);

void VBRsymm_MatMult_CaABbC_double(const int mb, const int n, const int kb, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, const double beta, double *c, const int ldc, const int ind_base);

void VBRskew_MatMult_CaABbC_double(const int mb, const int n, const int kb, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, const double beta, double *c, const int ldc, const int ind_base);

void VBRskew_MatMult_CaATBbC_double(const int mb, const int n, const int kb, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, const double beta, double *c, const int ldc, const int ind_base);

4.2 Variable Block Row Vector Multiply Routines

void VBR_VecMult_CAB_double(const int mb, const int kb, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, double *c, const int ind_base);

void VBR_VecMult_CATB_double(const int mb, const int kb, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, double *c, const int ind_base);

void VBRsymm_VecMult_CAB_double(const int mb, const int kb, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, double *c, const int ind_base);

void VBRskew_VecMult_CAB_double(const int mb, const int kb, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, double *c, const int ind_base);

void VBRskew_VecMult_CATB_double(const int mb, const int kb, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, double *c, const int ind_base);

void VBR_VecMult_CaAB_double(const int mb, const int kb, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, double *c, const int ind_base);

void VBR_VecMult_CaATB_double(const int mb, const int kb, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, double *c, const int ind_base);

void VBRsymm_VecMult_CaAB_double(const int mb, const int kb, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, double *c, const int ind_base);

void VBRskew_VecMult_CaAB_double(const int mb, const int kb, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, double *c, const int ind_base);

void VBRskew_VecMult_CaATB_double(const int mb, const int kb, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, double *c, const int ind_base);

void VBR_VecMult_CABC_double(const int mb, const int kb, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, double *c, const int ind_base);

void VBR_VecMult_CATBC_double(const int mb, const int kb, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, double *c, const int ind_base);

void VBRsymm_VecMult_CABC_double(const int mb, const int kb, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, double *c, const int ind_base);

void VBRskew_VecMult_CABC_double(const int mb, const int kb, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, double *c, const int ind_base);

void VBRskew_VecMult_CATBC_double(const int mb, const int kb, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, double *c, const int ind_base);

void VBR_VecMult_CaABC_double(const int mb, const int kb, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, double *c, const int ind_base);

void VBR_VecMult_CaATBC_double(const int mb, const int kb, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, double *c, const int ind_base);

void VBRsymm_VecMult_CaABC_double(const int mb, const int kb, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, double *c, const int ind_base);

void VBRskew_VecMult_CaABC_double(const int mb, const int kb, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, double *c, const int ind_base);

void VBRskew_VecMult_CaATBC_double(const int mb, const int kb, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, double *c, const int ind_base);

void VBR_VecMult_CABbC_double(const int mb, const int kb, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const double beta, double *c, const int ind_base);

void VBR_VecMult_CATBbC_double(const int mb, const int kb, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const double beta, double *c, const int ind_base);

void VBRsymm_VecMult_CABbC_double(const int mb, const int kb, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const double beta, double *c, const int ind_base);

void VBRskew_VecMult_CABbC_double(const int mb, const int kb, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const double beta, double *c, const int ind_base);

void VBRskew_VecMult_CATBbC_double(const int mb, const int kb, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const double beta, double *c, const int ind_base);

void VBR_VecMult_CaABbC_double(const int mb, const int kb, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const double beta, double *c, const int ind_base);

void VBR_VecMult_CaATBbC_double(const int mb, const int kb, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const double beta, double *c, const int ind_base);

void VBRsymm_VecMult_CaABbC_double(const int mb, const int kb, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const double beta, double *c, const int ind_base);

void VBRskew_VecMult_CaABbC_double(const int mb, const int kb, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const double beta, double *c, const int ind_base);

void VBRskew_VecMult_CaATBbC_double(const int mb, const int kb, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const double beta, double *c, const int ind_base);

4.3 Variable Block Row Matrix Triangular Solve Routines

void VBR_MatTriangSlvLU_CAB_double(const int mb, const int n, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, double *c, const int ldc, const int ind_base);

void VBR_MatTriangSlvUU_CAB_double(const int mb, const int n, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, double *c, const int ldc, const int ind_base);

void VBR_MatTriangSlvLU_CATB_double(const int mb, const int n, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, double *c, const int ldc, const int ind_base);

void VBR_MatTriangSlvUU_CATB_double(const int mb, const int n, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, double *c, const int ldc, const int ind_base);

void VBR_MatTriangSlvLU_CaAB_double(const int mb, const int n, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, double *c, const int ldc, const int ind_base);

void VBR_MatTriangSlvUU_CaAB_double(const int mb, const int n, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, double *c, const int ldc, const int ind_base);

void VBR_MatTriangSlvLU_CaATB_double(const int mb, const int n, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, double *c, const int ldc, const int ind_base);

void VBR_MatTriangSlvUU_CaATB_double(const int mb, const int n, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, double *c, const int ldc, const int ind_base);

void VBR_MatTriangSlvLU_CABC_double(const int mb, const int n, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, double *c, const int ldc, double *work, const int ind_base);

void VBR_MatTriangSlvUU_CABC_double(const int mb, const int n, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, double *c, const int ldc, double *work, const int ind_base);

void VBR_MatTriangSlvLU_CATBC_double(const int mb, const int n, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, double *c, const int ldc, double *work, const int ind_base);

void VBR_MatTriangSlvUU_CATBC_double(const int mb, const int n, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, double *c, const int ldc, double *work, const int ind_base);

void VBR_MatTriangSlvLU_CaABC_double(const int mb, const int n, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, double *c, const int ldc, double *work, const int ind_base);

void VBR_MatTriangSlvUU_CaABC_double(const int mb, const int n, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, double *c, const int ldc, double *work, const int ind_base);

void VBR_MatTriangSlvLU_CaATBC_double(const int mb, const int n, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, double *c, const int ldc, double *work, const int ind_base);

void VBR_MatTriangSlvUU_CaATBC_double(const int mb, const int n, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, double *c, const int ldc, double *work, const int ind_base);

void VBR_MatTriangSlvLU_CABbC_double(const int mb, const int n, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, const double beta, double *c, const int ldc, double *work, const int ind_base);

void VBR_MatTriangSlvUU_CABbC_double(const int mb, const int n, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, const double beta, double *c, const int ldc, double *work, const int ind_base);

void VBR_MatTriangSlvLU_CATBbC_double(const int mb, const int n, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, const double beta, double *c, const int ldc, double *work, const int ind_base);

void VBR_MatTriangSlvUU_CATBbC_double(const int mb, const int n, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, const double beta, double *c, const int ldc, double *work, const int ind_base);

void VBR_MatTriangSlvLU_CaABbC_double(const int mb, const int n, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, const double beta, double *c, const int ldc, double *work, const int ind_base);

void VBR_MatTriangSlvUU_CaABbC_double(const int mb, const int n, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, const double beta, double *c, const int ldc, double *work, const int ind_base);

void VBR_MatTriangSlvLU_CaATBbC_double(const int mb, const int n, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, const double beta, double *c, const int ldc, double *work, const int ind_base);

void VBR_MatTriangSlvUU_CaATBbC_double(const int mb, const int n, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, const double beta, double *c, const int ldc, double *work, const int ind_base);

void VBR_MatTriangSlvLU_CDAB_double(const int mb, const int n, const double *dvl, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, double *c, const int ldc, double *work, const int ind_base);

void VBR_MatTriangSlvUU_CDAB_double(const int mb, const int n, const double *dvl, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, double *c, const int ldc, double *work, const int ind_base);

void VBR_MatTriangSlvLU_CDATB_double(const int mb, const int n, const double *dvl, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, double *c, const int ldc, double *work, const int ind_base);

void VBR_MatTriangSlvUU_CDATB_double(const int mb, const int n, const double *dvl, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, double *c, const int ldc, double *work, const int ind_base);

void VBR_MatTriangSlvLU_CaDAB_double(const int mb, const int n, const double *dvl, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, double *c, const int ldc, double *work, const int ind_base);

void VBR_MatTriangSlvUU_CaDAB_double(const int mb, const int n, const double *dvl, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, double *c, const int ldc, double *work, const int ind_base);

void VBR_MatTriangSlvLU_CaDATB_double(const int mb, const int n, const double *dvl, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, double *c, const int ldc, double *work, const int ind_base);

void VBR_MatTriangSlvUU_CaDATB_double(const int mb, const int n, const double *dvl, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, double *c, const int ldc, double *work, const int ind_base);

void VBR_MatTriangSlvLU_CDABC_double(const int mb, const int n, const double *dvl, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, double *c, const int ldc, double *work, const int ind_base);

void VBR_MatTriangSlvUU_CDABC_double(const int mb, const int n, const double *dvl, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, double *c, const int ldc, double *work, const int ind_base);

void VBR_MatTriangSlvLU_CDATBC_double(const int mb, const int n, const double *dvl, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, double *c, const int ldc, double *work, const int ind_base);

void VBR_MatTriangSlvUU_CDATBC_double(const int mb, const int n, const double *dvl, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, double *c, const int ldc, double *work, const int ind_base);

void VBR_MatTriangSlvLU_CaDABC_double(const int mb, const int n, const double *dvl, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, double *c, const int ldc, double *work, const int ind_base);

void VBR_MatTriangSlvUU_CaDABC_double(const int mb, const int n, const double *dvl, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, double *c, const int ldc, double *work, const int ind_base);

void VBR_MatTriangSlvLU_CaDATBC_double(const int mb, const int n, const double *dvl, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, double *c, const int ldc, double *work, const int ind_base);

void VBR_MatTriangSlvUU_CaDATBC_double(const int mb, const int n, const double *dvl, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, double *c, const int ldc, double *work, const int ind_base);

void VBR_MatTriangSlvLU_CDABbC_double(const int mb, const int n, const double *dvl, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, const double beta, double *c, const int ldc, double *work, const int ind_base);

void VBR_MatTriangSlvUU_CDABbC_double(const int mb, const int n, const double *dvl, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, const double beta, double *c, const int ldc, double *work, const int ind_base);

void VBR_MatTriangSlvLU_CDATBbC_double(const int mb, const int n, const double *dvl, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, const double beta, double *c, const int ldc, double *work, const int ind_base);

void VBR_MatTriangSlvUU_CDATBbC_double(const int mb, const int n, const double *dvl, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, const double beta, double *c, const int ldc, double *work, const int ind_base);

void VBR_MatTriangSlvLU_CaDABbC_double(const int mb, const int n, const double *dvl, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, const double beta, double *c, const int ldc, double *work, const int ind_base);

void VBR_MatTriangSlvUU_CaDABbC_double(const int mb, const int n, const double *dvl, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, const double beta, double *c, const int ldc, double *work, const int ind_base);

void VBR_MatTriangSlvLU_CaDATBbC_double(const int mb, const int n, const double *dvl, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, const double beta, double *c, const int ldc, double *work, const int ind_base);

void VBR_MatTriangSlvUU_CaDATBbC_double(const int mb, const int n, const double *dvl, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, const double beta, double *c, const int ldc, double *work, const int ind_base);

void VBR_MatTriangSlvLU_CADB_double(const int mb, const int n, const double *dvr, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, double *c, const int ldc, const int ind_base);

void VBR_MatTriangSlvUU_CADB_double(const int mb, const int n, const double *dvr, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, double *c, const int ldc, const int ind_base);

void VBR_MatTriangSlvLU_CATDB_double(const int mb, const int n, const double *dvr, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, double *c, const int ldc, const int ind_base);

void VBR_MatTriangSlvUU_CATDB_double(const int mb, const int n, const double *dvr, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, double *c, const int ldc, const int ind_base);

void VBR_MatTriangSlvLU_CaADB_double(const int mb, const int n, const double *dvr, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, double *c, const int ldc, const int ind_base);

void VBR_MatTriangSlvUU_CaADB_double(const int mb, const int n, const double *dvr, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, double *c, const int ldc, const int ind_base);

void VBR_MatTriangSlvLU_CaATDB_double(const int mb, const int n, const double *dvr, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, double *c, const int ldc, const int ind_base);

void VBR_MatTriangSlvUU_CaATDB_double(const int mb, const int n, const double *dvr, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, double *c, const int ldc, const int ind_base);

void VBR_MatTriangSlvLU_CADBC_double(const int mb, const int n, const double *dvr, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, double *c, const int ldc, double *work, const int ind_base);

void VBR_MatTriangSlvUU_CADBC_double(const int mb, const int n, const double *dvr, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, double *c, const int ldc, double *work, const int ind_base);

void VBR_MatTriangSlvLU_CATDBC_double(const int mb, const int n, const double *dvr, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, double *c, const int ldc, double *work, const int ind_base);

void VBR_MatTriangSlvUU_CATDBC_double(const int mb, const int n, const double *dvr, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, double *c, const int ldc, double *work, const int ind_base);

void VBR_MatTriangSlvLU_CaADBC_double(const int mb, const int n, const double *dvr, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, double *c, const int ldc, double *work, const int ind_base);

void VBR_MatTriangSlvUU_CaADBC_double(const int mb, const int n, const double *dvr, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, double *c, const int ldc, double *work, const int ind_base);

void VBR_MatTriangSlvLU_CaATDBC_double(const int mb, const int n, const double *dvr, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, double *c, const int ldc, double *work, const int ind_base);

void VBR_MatTriangSlvUU_CaATDBC_double(const int mb, const int n, const double *dvr, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, double *c, const int ldc, double *work, const int ind_base);

void VBR_MatTriangSlvLU_CADBbC_double(const int mb, const int n, const double *dvr, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, const double beta, double *c, const int ldc, double *work, const int ind_base);

void VBR_MatTriangSlvUU_CADBbC_double(const int mb, const int n, const double *dvr, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, const double beta, double *c, const int ldc, double *work, const int ind_base);

void VBR_MatTriangSlvLU_CATDBbC_double(const int mb, const int n, const double *dvr, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, const double beta, double *c, const int ldc, double *work, const int ind_base);

void VBR_MatTriangSlvUU_CATDBbC_double(const int mb, const int n, const double *dvr, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, const double beta, double *c, const int ldc, double *work, const int ind_base);

void VBR_MatTriangSlvLU_CaADBbC_double(const int mb, const int n, const double *dvr, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, const double beta, double *c, const int ldc, double *work, const int ind_base);

void VBR_MatTriangSlvUU_CaADBbC_double(const int mb, const int n, const double *dvr, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, const double beta, double *c, const int ldc, double *work, const int ind_base);

void VBR_MatTriangSlvLU_CaATDBbC_double(const int mb, const int n, const double *dvr, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, const double beta, double *c, const int ldc, double *work, const int ind_base);

void VBR_MatTriangSlvUU_CaATDBbC_double(const int mb, const int n, const double *dvr, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, const double beta, double *c, const int ldc, double *work, const int ind_base);

void VBR_MatTriangSlvLU_CDADB_double(const int mb, const int n, const double *dvl, const double *dvr, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, double *c, const int ldc, double *work, const int ind_base);

void VBR_MatTriangSlvUU_CDADB_double(const int mb, const int n, const double *dvl, const double *dvr, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, double *c, const int ldc, double *work, const int ind_base);

void VBR_MatTriangSlvLU_CDATDB_double(const int mb, const int n, const double *dvl, const double *dvr, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, double *c, const int ldc, double *work, const int ind_base);

void VBR_MatTriangSlvUU_CDATDB_double(const int mb, const int n, const double *dvl, const double *dvr, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, double *c, const int ldc, double *work, const int ind_base);

void VBR_MatTriangSlvLU_CaDADB_double(const int mb, const int n, const double *dvl, const double *dvr, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, double *c, const int ldc, double *work, const int ind_base);

void VBR_MatTriangSlvUU_CaDADB_double(const int mb, const int n, const double *dvl, const double *dvr, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, double *c, const int ldc, double *work, const int ind_base);

void VBR_MatTriangSlvLU_CaDATDB_double(const int mb, const int n, const double *dvl, const double *dvr, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, double *c, const int ldc, double *work, const int ind_base);

void VBR_MatTriangSlvUU_CaDATDB_double(const int mb, const int n, const double *dvl, const double *dvr, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, double *c, const int ldc, double *work, const int ind_base);

void VBR_MatTriangSlvLU_CDADBC_double(const int mb, const int n, const double *dvl, const double *dvr, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, double *c, const int ldc, double *work, const int ind_base);

void VBR_MatTriangSlvUU_CDADBC_double(const int mb, const int n, const double *dvl, const double *dvr, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, double *c, const int ldc, double *work, const int ind_base);

void VBR_MatTriangSlvLU_CDATDBC_double(const int mb, const int n, const double *dvl, const double *dvr, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, double *c, const int ldc, double *work, const int ind_base);

void VBR_MatTriangSlvUU_CDATDBC_double(const int mb, const int n, const double *dvl, const double *dvr, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, double *c, const int ldc, double *work, const int ind_base);

void VBR_MatTriangSlvLU_CaDADBC_double(const int mb, const int n, const double *dvl, const double *dvr, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, double *c, const int ldc, double *work, const int ind_base);

void VBR_MatTriangSlvUU_CaDADBC_double(const int mb, const int n, const double *dvl, const double *dvr, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, double *c, const int ldc, double *work, const int ind_base);

void VBR_MatTriangSlvLU_CaDATDBC_double(const int mb, const int n, const double *dvl, const double *dvr, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, double *c, const int ldc, double *work, const int ind_base);

void VBR_MatTriangSlvUU_CaDATDBC_double(const int mb, const int n, const double *dvl, const double *dvr, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, double *c, const int ldc, double *work, const int ind_base);

void VBR_MatTriangSlvLU_CDADBbC_double(const int mb, const int n, const double *dvl, const double *dvr, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, const double beta, double *c, const int ldc, double *work, const int ind_base);

void VBR_MatTriangSlvUU_CDADBbC_double(const int mb, const int n, const double *dvl, const double *dvr, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, const double beta, double *c, const int ldc, double *work, const int ind_base);

void VBR_MatTriangSlvLU_CDATDBbC_double(const int mb, const int n, const double *dvl, const double *dvr, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, const double beta, double *c, const int ldc, double *work, const int ind_base);

void VBR_MatTriangSlvUU_CDATDBbC_double(const int mb, const int n, const double *dvl, const double *dvr, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, const double beta, double *c, const int ldc, double *work, const int ind_base);

void VBR_MatTriangSlvLU_CaDADBbC_double(const int mb, const int n, const double *dvl, const double *dvr, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, const double beta, double *c, const int ldc, double *work, const int ind_base);

void VBR_MatTriangSlvUU_CaDADBbC_double(const int mb, const int n, const double *dvl, const double *dvr, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, const double beta, double *c, const int ldc, double *work, const int ind_base);

void VBR_MatTriangSlvLU_CaDATDBbC_double(const int mb, const int n, const double *dvl, const double *dvr, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, const double beta, double *c, const int ldc, double *work, const int ind_base);

void VBR_MatTriangSlvUU_CaDATDBbC_double(const int mb, const int n, const double *dvl, const double *dvr, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const int ldb, const double beta, double *c, const int ldc, double *work, const int ind_base);

4.4 Variable Block Row Vector Triangular Solve Routines

void VBR_VecTriangSlvLU_CAB_double(const int mb, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, double *c, const int ind_base);

void VBR_VecTriangSlvUU_CAB_double(const int mb, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, double *c, const int ind_base);

void VBR_VecTriangSlvLU_CATB_double(const int mb, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, double *c, const int ind_base);

void VBR_VecTriangSlvUU_CATB_double(const int mb, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, double *c, const int ind_base);

void VBR_VecTriangSlvLU_CaAB_double(const int mb, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, double *c, const int ind_base);

void VBR_VecTriangSlvUU_CaAB_double(const int mb, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, double *c, const int ind_base);

void VBR_VecTriangSlvLU_CaATB_double(const int mb, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, double *c, const int ind_base);

void VBR_VecTriangSlvUU_CaATB_double(const int mb, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, double *c, const int ind_base);

void VBR_VecTriangSlvLU_CABC_double(const int mb, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, double *c, double *work, const int ind_base);

void VBR_VecTriangSlvUU_CABC_double(const int mb, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, double *c, double *work, const int ind_base);

void VBR_VecTriangSlvLU_CATBC_double(const int mb, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, double *c, double *work, const int ind_base);

void VBR_VecTriangSlvUU_CATBC_double(const int mb, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, double *c, double *work, const int ind_base);

void VBR_VecTriangSlvLU_CaABC_double(const int mb, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, double *c, double *work, const int ind_base);

void VBR_VecTriangSlvUU_CaABC_double(const int mb, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, double *c, double *work, const int ind_base);

void VBR_VecTriangSlvLU_CaATBC_double(const int mb, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, double *c, double *work, const int ind_base);

void VBR_VecTriangSlvUU_CaATBC_double(const int mb, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, double *c, double *work, const int ind_base);

void VBR_VecTriangSlvLU_CABbC_double(const int mb, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const double beta, double *c, double *work, const int ind_base);

void VBR_VecTriangSlvUU_CABbC_double(const int mb, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const double beta, double *c, double *work, const int ind_base);

void VBR_VecTriangSlvLU_CATBbC_double(const int mb, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const double beta, double *c, double *work, const int ind_base);

void VBR_VecTriangSlvUU_CATBbC_double(const int mb, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const double beta, double *c, double *work, const int ind_base);

void VBR_VecTriangSlvLU_CaABbC_double(const int mb, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const double beta, double *c, double *work, const int ind_base);

void VBR_VecTriangSlvUU_CaABbC_double(const int mb, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const double beta, double *c, double *work, const int ind_base);

void VBR_VecTriangSlvLU_CaATBbC_double(const int mb, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const double beta, double *c, double *work, const int ind_base);

void VBR_VecTriangSlvUU_CaATBbC_double(const int mb, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const double beta, double *c, double *work, const int ind_base);

void VBR_VecTriangSlvLU_CDAB_double(const int mb, const double *dvl, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, double *c, double *work, const int ind_base);

void VBR_VecTriangSlvUU_CDAB_double(const int mb, const double *dvl, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, double *c, double *work, const int ind_base);

void VBR_VecTriangSlvLU_CDATB_double(const int mb, const double *dvl, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, double *c, double *work, const int ind_base);

void VBR_VecTriangSlvUU_CDATB_double(const int mb, const double *dvl, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, double *c, double *work, const int ind_base);

void VBR_VecTriangSlvLU_CaDAB_double(const int mb, const double *dvl, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, double *c, double *work, const int ind_base);

void VBR_VecTriangSlvUU_CaDAB_double(const int mb, const double *dvl, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, double *c, double *work, const int ind_base);

void VBR_VecTriangSlvLU_CaDATB_double(const int mb, const double *dvl, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, double *c, double *work, const int ind_base);

void VBR_VecTriangSlvUU_CaDATB_double(const int mb, const double *dvl, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, double *c, double *work, const int ind_base);

void VBR_VecTriangSlvLU_CDABC_double(const int mb, const double *dvl, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, double *c, double *work, const int ind_base);

void VBR_VecTriangSlvUU_CDABC_double(const int mb, const double *dvl, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, double *c, double *work, const int ind_base);

void VBR_VecTriangSlvLU_CDATBC_double(const int mb, const double *dvl, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, double *c, double *work, const int ind_base);

void VBR_VecTriangSlvUU_CDATBC_double(const int mb, const double *dvl, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, double *c, double *work, const int ind_base);

void VBR_VecTriangSlvLU_CaDABC_double(const int mb, const double *dvl, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, double *c, double *work, const int ind_base);

void VBR_VecTriangSlvUU_CaDABC_double(const int mb, const double *dvl, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, double *c, double *work, const int ind_base);

void VBR_VecTriangSlvLU_CaDATBC_double(const int mb, const double *dvl, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, double *c, double *work, const int ind_base);

void VBR_VecTriangSlvUU_CaDATBC_double(const int mb, const double *dvl, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, double *c, double *work, const int ind_base);

void VBR_VecTriangSlvLU_CDABbC_double(const int mb, const double *dvl, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const double beta, double *c, double *work, const int ind_base);

void VBR_VecTriangSlvUU_CDABbC_double(const int mb, const double *dvl, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const double beta, double *c, double *work, const int ind_base);

void VBR_VecTriangSlvLU_CDATBbC_double(const int mb, const double *dvl, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const double beta, double *c, double *work, const int ind_base);

void VBR_VecTriangSlvUU_CDATBbC_double(const int mb, const double *dvl, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const double beta, double *c, double *work, const int ind_base);

void VBR_VecTriangSlvLU_CaDABbC_double(const int mb, const double *dvl, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const double beta, double *c, double *work, const int ind_base);

void VBR_VecTriangSlvUU_CaDABbC_double(const int mb, const double *dvl, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const double beta, double *c, double *work, const int ind_base);

void VBR_VecTriangSlvLU_CaDATBbC_double(const int mb, const double *dvl, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const double beta, double *c, double *work, const int ind_base);

void VBR_VecTriangSlvUU_CaDATBbC_double(const int mb, const double *dvl, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const double beta, double *c, double *work, const int ind_base);

void VBR_VecTriangSlvLU_CADB_double(const int mb, const double *dvr, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, double *c, const int ind_base);

void VBR_VecTriangSlvUU_CADB_double(const int mb, const double *dvr, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, double *c, const int ind_base);

void VBR_VecTriangSlvLU_CATDB_double(const int mb, const double *dvr, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, double *c, const int ind_base);

void VBR_VecTriangSlvUU_CATDB_double(const int mb, const double *dvr, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, double *c, const int ind_base);

void VBR_VecTriangSlvLU_CaADB_double(const int mb, const double *dvr, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, double *c, const int ind_base);

void VBR_VecTriangSlvUU_CaADB_double(const int mb, const double *dvr, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, double *c, const int ind_base);

void VBR_VecTriangSlvLU_CaATDB_double(const int mb, const double *dvr, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, double *c, const int ind_base);

void VBR_VecTriangSlvUU_CaATDB_double(const int mb, const double *dvr, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, double *c, const int ind_base);

void VBR_VecTriangSlvLU_CADBC_double(const int mb, const double *dvr, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, double *c, double *work, const int ind_base);

void VBR_VecTriangSlvUU_CADBC_double(const int mb, const double *dvr, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, double *c, double *work, const int ind_base);

void VBR_VecTriangSlvLU_CATDBC_double(const int mb, const double *dvr, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, double *c, double *work, const int ind_base);

void VBR_VecTriangSlvUU_CATDBC_double(const int mb, const double *dvr, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, double *c, double *work, const int ind_base);

void VBR_VecTriangSlvLU_CaADBC_double(const int mb, const double *dvr, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, double *c, double *work, const int ind_base);

void VBR_VecTriangSlvUU_CaADBC_double(const int mb, const double *dvr, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, double *c, double *work, const int ind_base);

void VBR_VecTriangSlvLU_CaATDBC_double(const int mb, const double *dvr, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, double *c, double *work, const int ind_base);

void VBR_VecTriangSlvUU_CaATDBC_double(const int mb, const double *dvr, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, double *c, double *work, const int ind_base);

void VBR_VecTriangSlvLU_CADBbC_double(const int mb, const double *dvr, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const double beta, double *c, double *work, const int ind_base);

void VBR_VecTriangSlvUU_CADBbC_double(const int mb, const double *dvr, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const double beta, double *c, double *work, const int ind_base);

void VBR_VecTriangSlvLU_CATDBbC_double(const int mb, const double *dvr, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const double beta, double *c, double *work, const int ind_base);

void VBR_VecTriangSlvUU_CATDBbC_double(const int mb, const double *dvr, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const double beta, double *c, double *work, const int ind_base);

void VBR_VecTriangSlvLU_CaADBbC_double(const int mb, const double *dvr, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const double beta, double *c, double *work, const int ind_base);

void VBR_VecTriangSlvUU_CaADBbC_double(const int mb, const double *dvr, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const double beta, double *c, double *work, const int ind_base);

void VBR_VecTriangSlvLU_CaATDBbC_double(const int mb, const double *dvr, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const double beta, double *c, double *work, const int ind_base);

void VBR_VecTriangSlvUU_CaATDBbC_double(const int mb, const double *dvr, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const double beta, double *c, double *work, const int ind_base);

void VBR_VecTriangSlvLU_CDADB_double(const int mb, const double *dvl, const double *dvr, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, double *c, double *work, const int ind_base);

void VBR_VecTriangSlvUU_CDADB_double(const int mb, const double *dvl, const double *dvr, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, double *c, double *work, const int ind_base);

void VBR_VecTriangSlvLU_CDATDB_double(const int mb, const double *dvl, const double *dvr, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, double *c, double *work, const int ind_base);

void VBR_VecTriangSlvUU_CDATDB_double(const int mb, const double *dvl, const double *dvr, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, double *c, double *work, const int ind_base);

void VBR_VecTriangSlvLU_CaDADB_double(const int mb, const double *dvl, const double *dvr, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, double *c, double *work, const int ind_base);

void VBR_VecTriangSlvUU_CaDADB_double(const int mb, const double *dvl, const double *dvr, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, double *c, double *work, const int ind_base);

void VBR_VecTriangSlvLU_CaDATDB_double(const int mb, const double *dvl, const double *dvr, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, double *c, double *work, const int ind_base);

void VBR_VecTriangSlvUU_CaDATDB_double(const int mb, const double *dvl, const double *dvr, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, double *c, double *work, const int ind_base);

void VBR_VecTriangSlvLU_CDADBC_double(const int mb, const double *dvl, const double *dvr, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, double *c, double *work, const int ind_base);

void VBR_VecTriangSlvUU_CDADBC_double(const int mb, const double *dvl, const double *dvr, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, double *c, double *work, const int ind_base);

void VBR_VecTriangSlvLU_CDATDBC_double(const int mb, const double *dvl, const double *dvr, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, double *c, double *work, const int ind_base);

void VBR_VecTriangSlvUU_CDATDBC_double(const int mb, const double *dvl, const double *dvr, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, double *c, double *work, const int ind_base);

void VBR_VecTriangSlvLU_CaDADBC_double(const int mb, const double *dvl, const double *dvr, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, double *c, double *work, const int ind_base);

void VBR_VecTriangSlvUU_CaDADBC_double(const int mb, const double *dvl, const double *dvr, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, double *c, double *work, const int ind_base);

void VBR_VecTriangSlvLU_CaDATDBC_double(const int mb, const double *dvl, const double *dvr, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, double *c, double *work, const int ind_base);

void VBR_VecTriangSlvUU_CaDATDBC_double(const int mb, const double *dvl, const double *dvr, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, double *c, double *work, const int ind_base);

void VBR_VecTriangSlvLU_CDADBbC_double(const int mb, const double *dvl, const double *dvr, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const double beta, double *c, double *work, const int ind_base);

void VBR_VecTriangSlvUU_CDADBbC_double(const int mb, const double *dvl, const double *dvr, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const double beta, double *c, double *work, const int ind_base);

void VBR_VecTriangSlvLU_CDATDBbC_double(const int mb, const double *dvl, const double *dvr, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const double beta, double *c, double *work, const int ind_base);

void VBR_VecTriangSlvUU_CDATDBbC_double(const int mb, const double *dvl, const double *dvr, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const double beta, double *c, double *work, const int ind_base);

void VBR_VecTriangSlvLU_CaDADBbC_double(const int mb, const double *dvl, const double *dvr, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const double beta, double *c, double *work, const int ind_base);

void VBR_VecTriangSlvUU_CaDADBbC_double(const int mb, const double *dvl, const double *dvr, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const double beta, double *c, double *work, const int ind_base);

void VBR_VecTriangSlvLU_CaDATDBbC_double(const int mb, const double *dvl, const double *dvr, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const double beta, double *c, double *work, const int ind_base);

void VBR_VecTriangSlvUU_CaDATDBbC_double(const int mb, const double *dvl, const double *dvr, const double alpha, const double *val, const int *indx, const int *bindx, const int *rpntr, const int *cpntr, const int *bpntrb, const int *bpntre, const double *b, const double beta, double *c, double *work, const int ind_base);

A Installation Instructions

The installation of the Sparse BLAS Toolkit is automated with the "make" utility. To use "make" to build the library:

1. Edit the file ./makefile.def to reflect your system setup:
   - The minimum installation requires an ANSI C compiler.
   - An extended installation which includes Fortran callable routines and testers is available. If the presence of a Fortran compiler is indicated in the makefile.def file, the extended version will be installed.
   - The archival process by default uses "ranlib". If this is not available on your system, set HASRANLIB to 'f'.
2. Type:

make install *	to build the library AND make and run the C and Fortran testers
make installc	to build the library AND make and run the C testers
make library	to build the archive file ./lib/libsptk.a (tests are not built)
make testc	to build and run the C testers (library must be pre-built)
make testf77 *	to build and run the Fortran testers (library must be pre-built)

   * requires a Fortran compiler

3. For space-saving cleanup, type "make clean" to remove all .o

B References

[1] S. Carney, M. Heroux, G. Li, R. Pozo, K. Remington, and K. Wu, *A revised proposal for a sparse blas toolkit*, http://www.cs.sandia.gov/ mheroux/PS/spblastk.ps, (1996).

[2] I. Duff, M. Marrone, and G. Radicati, *A proposal for user level sparse blas*, Tech. Report TR/PA/92/85, CERFACS, Toulouse, France, 1992.