

**Parallel Linear Algebra  
for Scalable Multi-core Architectures  
(PLASMA)**

**Reference Manual**

2008-11-15

## 1. Mathematic function prototypes

### (a) DPOSV

This routine computes the solution to a real system of linear equations  $A \times X = B$ , where  $A$  is an N-by-N symmetric positive definite matrix and  $X$  and  $B$  are N-by-NRHS matrices.

The Cholesky decomposition is used to factor  $A$  as  $A = U^T \times U$ , if UPLO = 'U', or  $A = L \times L^T$ , if UPLO = 'L', where  $U$  is an upper triangular matrix and  $L$  is a lower triangular matrix. The factored form of  $A$  is then used to solve the system of equations  $A \times X = B$ .

#### C binding:

```
int plasma_DPOSV(PLASMA_enum UPLO, int N, int NRHS,
                 double *A, int LDA, double *B, int LDB)
```

#### Fortran binding:

```
void PLASMA_DPOSV(UPLO, N, NRHS, A, LDA, B, LDB, INFO)
  INTEGER UPLO, N, NRHS, LDA, LDB, INFO
  DOUBLE PRECISION A(LDA, *), B(LDB, *)
```

### (b) DPOTRF

DPOTRF computes the Cholesky factorization of a real symmetric positive definite matrix  $A$ .

The factorization has the form  $A = U^T \times U$ , if UPLO = 'U', or  $A = L \times L^T$ , if UPLO = 'L', where  $U$  is an upper triangular matrix and  $L$  is lower triangular.

#### C binding:

```
int plasma_DPOTRF(PLASMA_enum UPLO, int N,
                 double *A, int LDA)
```

#### Fortran binding:

```
void PLASMA_DPOTRF(UPLO, N, A, LDA, INFO)
  INTEGER UPLO, N, LDA, INFO
  DOUBLE PRECISION A(LDA, *)
```

(c) DPOTRS

DPOTRS solves a system of linear equations  $A \times X = B$  with a symmetric positive definite matrix  $A$  using the Cholesky factorization  $A = U^T \times U$  or  $A = L \times L^T$  computed by DPOTRF.

**C binding:**

```
int plasma_DPOTRS(PLASMA_enum UPLO, int N, int NRHS,
                  double *A, int LDA, double *B, int LDB)
```

**Fortran binding:**

```
void PLASMA_DPOTRS(UPLO, N, NRHS, A, LDA, B, LDB, INFO)
    INTEGER UPLO, N, NRHS, LDA, LDB, INFO
    DOUBLE PRECISION A(LDA, *), B(LDB, *)
```

(d) DTRSM

DTRSM solves the matrix equation  $op(A) \times X = \alpha \times B$ , where  $\alpha$  is a scalar,  $X$  and  $B$  are M-by-N matrices,  $A$  is a unit, or non-unit, upper or lower triangular matrix and  $op(A)$  is one of  $op(A) = A$  or  $op(A) = A^T$ .

The matrix  $X$  is overwritten on  $B$ . This routine works only for SIDE='L'.

**C binding:**

```
int plasma_DTRSM(PLASMA_enum SIDE, PLASMA_enum UPLO,
                 PLASMA_enum TRANS, PLASMA_enum diag,
                 int N, int NRHS, double *A, int LDA,
                 double *B, int LDB)
```

**Fortran binding:**

```
void PLASMA_DTRSM(SIDE, UPLO, TRANS, DIAG, N, NRHS,
                  A, LDA, B, LDB, INFO)
    INTEGER SIDE, UPLO, TRANS, DIAG, N, NRHS,
    DOUBLE PRECISION A(LDA, *), B(LDB, *)
```

(e) DGELS

DGELS solves overdetermined real linear systems involving an M-by-N matrix  $A$  using a  $QR$  factorization of  $A$ . It is assumed that  $A$  has full rank.

The following options are provided:

- i. If  $M \geq N$ : find the least squares solution of an overdetermined system, i.e., solve the least squares problem minimize  $\|B - A \times X\|$ .

This routine works only for TRANS='N'.

**C binding:**

```
int plasma_DGELS(PLASMA_enum TRANS, int M, int N,
                 int NRHS, double *A, int LDA,
                 double *T, double *B, int LDB)
```

**Fortran binding:**

```
void PLASMA_DGELS(TRANS, M, N, NRHS, A, LDA, T,
                  B, LDB, INFO)
    INTEGER TRANS, DIAG, M, N, NRHS, LDA, LDB, INFO
    INTEGER*8 T
    DOUBLE PRECISION A(LDA, *), B(LDB, *)
```

(f) DGEQRF

DGEQRF computes a QR factorization of a real M-by-N matrix  $A = Q \times R$ .

**C binding:**

```
int plasma_DGEQRF(int M, int N, double *A, int LDA,
                  double *T)
```

**Fortran binding:**

```
void PLASMA_DGEQRF(M, N, A, LDA, T, INFO)
    INTEGER M, N, LDA, INFO
    INTEGER*8 T
    DOUBLE PRECISION A(LDA, *)
```

(g) DORMQR

DORMQR overwrites the general real M-by-N matrix  $B$  with

	$SIDE = 'L'$	$SIDE = 'R'$
$TRANS = N$	$Q \times B$	$B \times Q$
$TRANS = T$	$Q^T \times B$	$B \times Q^T$

where  $Q$  is a real matrix orthogonal by block as returned by DGE-QRF.  $Q$  is of order M if  $SIDE = 'L'$  and of order N if  $SIDE = 'R'$ . This routine works only for  $SIDE='L'$ .

**C binding:**

```
int plasma_DORMQR(PLASMA_enum SIDE, PLASMA_enum TRANS,
                  int M, int NRHS, int N, double *A, int LDA,
                  double *T, double *B, int LDB)
```

**Fortran binding:**

```
void PLASMA_DORMQR(SIDE, TRANS, M, NRHS, N, A, LDA,
                  T, B, LDB, INFO)
    INTEGER SIDE, TRANS, M, NRHS, N, LDA, LDB, INFO
    INTEGER*8 T
    DOUBLE PRECISION A(LDA, *), B(LDB, *)
```

(h) DGESV

DGESV computes the solution to a real system of linear equations  $A \times X = B$ , where  $A$  is an N-by-N matrix and  $X$  and  $B$  are N-by-NRHS matrices.

The  $LU$  decomposition with partial pivoting and row interchanges is used to factor  $A$  as  $A = P \times L \times U$ , where  $P$  is a permutation matrix,  $L$  is unit lower triangular, and  $U$  is upper triangular. The factored form of  $A$  is then used to solve the system of equations  $A \times X = B$ .

**C binding:**

```
int plasma_DGESV(int N, int NRHS, double *A, int LDA,
                  double *L, int *IPIV, double *B, int LDB)
```

**Fortran binding:**

```
void PLASMA_DGESV(N, NRHS, A, LDA, L, IPIV, B, LDB, INFO)
    INTEGER N, NRHS, LDA, LDB, INFO
```

```

    INTEGER*8 L, IPIV
    DOUBLE PRECISION A(LDA, *), B(LDB, *)

```

(i) DGETRF

DGETRF computes an LU factorization of a general M-by-N matrix  $A$  using partial pivoting with row interchanges.

The factorization has the form  $A = P \times L \times U$  where  $P$  is a permutation matrix,  $L$  is lower triangular with unit diagonal elements (lower trapezoidal if  $M > N$ ), and  $U$  is upper triangular (upper trapezoidal if  $M < N$ ).

**C binding:**

```

int plasma_DGETRF(int M, int N, double *A, int LDA,
                  double *L, int *IPIV)

```

**Fortran binding:**

```

void PLASMA_DGETRF(M, N, A, LDA, L, IPIV, INFO)
    INTEGER M, NRHS, N, LDA, INFO
    INTEGER*8 L, IPIV
    DOUBLE PRECISION A(LDA, *)

```

(j) DGETRS

DGETRS solves a system of linear equations  $A \times X = B$  with a general M-by-N matrix  $A$  and a M-by-NRHS  $B$  matrix using the LU factorization computed by DGETRF.

**C binding:**

```

int plasma_DGETRS(int M, int NRHS, int N, double *A,
                  int LDA, double *L, int *IPIV, double *B, int LDB)

```

**Fortran binding:**

```

void PLASMA_DGETRS(M, NRHS, N, A, LDA, L, IPIV,
                  B, LDB, INFO)
    INTEGER M, NRHS, N, LDA, LDB, INFO
    INTEGER*8 L, IPIV
    DOUBLE PRECISION A(LDA, *), B(LDB, *)

```

(k) DTRSMPL

DTRSMPL applies the factor  $L$  with the pivot IPIV from DGETRF to solve  $P \times L \times X = B$ , where  $L$  is an M-by-N matrix and X and B are N-by-NRHS matrices.

**C binding:**

```
int plasma_DTRSMPL(int M, int NRHS, int N, double *A,
    int LDA, double *L, int *IPIV, double *B, int LDB)
```

**Fortran binding:**

```
void PLASMA_DTRSMPL(M, NRHS, N,
    A, LDA, L, IPIV, B, LDB, INFO)
    INTEGER M, NRHS, N, LDA, LDB, INFO
    INTEGER*8 L, IPIV
    DOUBLE PRECISION A(LDA, *), B(LDB, *)
```

## 2. Auxiliary function prototypes

(a) Plasma\_Init

This routine checks internal hardware constraints and arranges PLASMA internal structures to fit the hardware, by e.g., probing the available number of cores, aligning memory to the cache line size, allocating memory in huge TLB pages, if available, etc.

**C binding:**

```
int plasma_Init(int M, int N, int NRHS)
```

**Fortran binding:**

```
void PLASMA_INIT(M, N, NRHS, INFO)
    INTEGER M, N, NRHS, INFO
```

(b) Plasma\_Finalize

This routine ends the parallel environment by joining and destroying the threads. Also, it releases any internal memory allocation needed during the execution.

**C binding:**

```
int plasma_Finalize()
```

**Fortran binding:**

```
void PLASMA_FINALIZE(INFO)
    INTEGER INFO
```

(c) Plasma\_Allocate\_T

This routine allocates the memory needed for the triangular factor T used in QR and LQ factorization.

**C binding:**

```
double *plasma_Allocate_T(int M, int N);
```

**Fortran binding:**

```
void PLASMA_ALLOCATE_T(T,M,N)
    INTEGER M, N
    INTEGER*8 T
```

(d) Plasma\_Free\_T

This routine frees the memory allocation of T.

**C binding:**

```
int plasma_Free_T(double *T)
```

**Fortran binding:**

```
void PLASMA_FREE_T(T, INFO)
    INTEGER INFO
    INTEGER*8 T
```

(e) Plasma\_Allocate\_L



This routine allocates the memory needed for the lower factor L used in LU factorization.

**C binding:**

```
double *plasma_Allocate_L(int M, int N)
```

**Fortran binding:**

```
void PLASMA_ALLOCATE_L(L, M, N)
    INTEGER M, N
    INTEGER*8 L
```

(f) Plasma\_Free\_L

This routine frees the memory allocation of the lower factor L.

**C binding:**

```
int plasma_Free_L(double *L)
```

**Fortran binding:**

```
void PLASMA_FREE_L(L, INFO)
    INTEGER INFO
    INTEGER*8 L
```

(g) Plasma\_Allocate\_IPIV

This routine allocates the memory needed for the pivot array IPIV used in LU factorization.

**C binding:**

```
int *plasma_Allocate_IPIV(int M, int N)
```

**Fortran binding:**

```
void PLASMA_ALLOCATE_IPIV(IPIV, M, N)
    INTEGER M, N
    INTEGER*8 IPIV
```

(h) Plasma\_Free\_IPIV

This routine frees the memory allocation of the pivot array IPIV.

**C binding:**

```
int plasma_Free_IPIV(double *IPIV)
```

**Fortran binding:**

```
void PLASMA_FREE_IPIV(IPIV, INFO)
  INTEGER INFO
  INTEGER*8 IPIV
```

### 3. Constants definition

Here is a list of the parameters defining the variables SIDE, UPLO, TRANS and DIAG.

<i>Name</i>	<i>Value</i>
<i>PlasmaNoTrans</i>	111
<i>PlasmaTrans</i>	112
<i>PlasmaConjTrans</i>	113
<i>PlasmaUpper</i>	121
<i>PlasmaLower</i>	122
<i>PlasmaNonUnit</i>	131
<i>PlasmaUnit</i>	132
<i>PlasmaLeft</i>	141
<i>PlasmaRight</i>	142
<i>PlasmaForward</i>	391
<i>PlasmaBackward</i>	392
<i>PlasmaColumnwise</i>	401
<i>PlasmaRowwise</i>	402