
COLLABORATORS

	<i>TITLE :</i>		
<i>ACTION</i>	<i>NAME</i>	<i>DATE</i>	<i>SIGNATURE</i>
WRITTEN BY		July 4, 2009	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

1	PLASMA_dgelqf	1
1.1	Purpose	1
1.2	Arguments	1
1.3	Return Value:	1
1.4	C Bindings	1
1.5	Fortran Bindings	1
1.6	Online Browsing	1
2	PLASMA_dgelqf_Tile	2
2.1	Purpose	2
2.2	Arguments	2
2.3	Return Value:	2
2.4	C Bindings	2
2.5	Fortran Bindings	2
2.6	Online Browsing	2
3	PLASMA_dgelqs	2
3.1	Purpose	2
3.2	Arguments	3
3.3	Return Value:	3
3.4	C Bindings	3
3.5	Fortran Bindings	3
3.6	Online Browsing	3
4	PLASMA_dgelqs_Tile	3
4.1	Purpose	3
4.2	Arguments	4
4.3	Return Value:	4
4.4	C Bindings	4
4.5	Fortran Bindings	4
4.6	Online Browsing	4
5	PLASMA_dgels	4
5.1	Purpose	4
5.2	Arguments	5
5.3	Return Value:	5
5.4	C Bindings	5
5.5	Fortran Bindings	6
5.6	Online Browsing	6

6	PLASMA_dgels_Tile	6
6.1	Purpose	6
6.2	Arguments	6
6.3	Return Value:	7
6.4	C Bindings	7
6.5	Fortran Bindings	7
6.6	Online Browsing	7
7	PLASMA_dgeqrf	7
7.1	Purpose	7
7.2	Arguments	7
7.3	Return Value:	8
7.4	C Bindings	8
7.5	Fortran Bindings	8
7.6	Online Browsing	8
8	PLASMA_dgeqrf_Tile	8
8.1	Purpose	8
8.2	Arguments	8
8.3	Return Value:	8
8.4	C Bindings	8
8.5	Fortran Bindings	9
8.6	Online Browsing	9
9	PLASMA_dgeqrs	9
9.1	Purpose	9
9.2	Arguments	9
9.3	Return Value:	9
9.4	C Bindings	9
9.5	Fortran Bindings	10
9.6	Online Browsing	10
10	PLASMA_dgeqrs_Tile	10
10.1	Purpose	10
10.2	Arguments	10
10.3	Return Value:	10
10.4	C Bindings	10
10.5	Fortran Bindings	10
10.6	Online Browsing	10

11 PLASMA_dgesv	11
11.1 Purpose	11
11.2 Arguments	11
11.3 Return Value:	11
11.4 C Bindings	11
11.5 Fortran Bindings	12
11.6 Online Browsing	12
12 PLASMA_dgesv_Tile	12
12.1 Purpose	12
12.2 Arguments	12
12.3 Return Value:	12
12.4 C Bindings	12
12.5 Fortran Bindings	12
12.6 Online Browsing	13
13 PLASMA_dgetrf	13
13.1 Purpose	13
13.2 Arguments	13
13.3 Return Value:	13
13.4 C Bindings	13
13.5 Fortran Bindings	13
13.6 Online Browsing	14
14 PLASMA_dgetrf_Tile	14
14.1 Purpose	14
14.2 Arguments	14
14.3 Return Value:	14
14.4 C Bindings	14
14.5 Fortran Bindings	14
14.6 Online Browsing	14
15 PLASMA_dgetrs	14
15.1 Purpose	14
15.2 Arguments	15
15.3 Return Value:	15
15.4 C Bindings	15
15.5 Fortran Bindings	15
15.6 Online Browsing	15

16 PLASMA_dgetrs_Tile	16
16.1 Purpose	16
16.2 Arguments	16
16.3 Return Value:	16
16.4 C Bindings	16
16.5 Fortran Bindings	16
16.6 Online Browsing	16
17 PLASMA_dposv	16
17.1 Purpose	16
17.2 Arguments	17
17.3 Return Value:	17
17.4 C Bindings	17
17.5 Fortran Bindings	18
17.6 Online Browsing	18
18 PLASMA_dposv_Tile	18
18.1 Purpose	18
18.2 Arguments	18
18.3 Return Value:	18
18.4 C Bindings	19
18.5 Fortran Bindings	19
18.6 Online Browsing	19
19 PLASMA_dpotrf	19
19.1 Purpose	19
19.2 Arguments	19
19.3 Return Value:	20
19.4 C Bindings	20
19.5 Fortran Bindings	20
19.6 Online Browsing	20
20 PLASMA_dpotrf_Tile	20
20.1 Purpose	20
20.2 Arguments	20
20.3 Return Value:	21
20.4 C Bindings	21
20.5 Fortran Bindings	21
20.6 Online Browsing	21

21 PLASMA_dpotrs	21
21.1 Purpose	21
21.2 Arguments	21
21.3 Return Value:	22
21.4 C Bindings	22
21.5 Fortran Bindings	22
21.6 Online Browsing	22
22 PLASMA_dpotrs_Tile	22
22.1 Purpose	22
22.2 Arguments	22
22.3 Return Value:	22
22.4 C Bindings	23
22.5 Fortran Bindings	23
22.6 Online Browsing	23
23 PLASMA_dsgesv	23
23.1 Purpose	23
23.2 Arguments	23
23.3 Return Value:	24
23.4 C Bindings	24
23.5 Fortran Bindings	24
23.6 Online Browsing	24
24 PLASMA_dsgesv_Tile	25
24.1 Purpose	25
24.2 Arguments	25
24.3 Return Value:	26
24.4 C Bindings	26
24.5 Fortran Bindings	26
24.6 Online Browsing	26
25 PLASMA_dtrsm	26
25.1 Purpose	26
25.2 Arguments	26
25.3 Return Value:	27
25.4 C Bindings	27
25.5 Fortran Bindings	27
25.6 Online Browsing	28

26 PLASMA_dtrsm_Tile	28
26.1 Purpose	28
26.2 Arguments	28
26.3 Return Value:	28
26.4 C Bindings	29
26.5 Fortran Bindings	29
26.6 Online Browsing	29
27 PLASMA_dtrsml	29
27.1 Purpose	29
27.2 Arguments	29
27.3 Return Value:	30
27.4 C Bindings	30
27.5 Fortran Bindings	30
27.6 Online Browsing	30
28 PLASMA_dtrsml_Tile	30
28.1 Purpose	30
28.2 Arguments	30
28.3 Return Value:	30
28.4 C Bindings	31
28.5 Fortran Bindings	31
28.6 Online Browsing	31
29 PLASMA_sgelqf	31
29.1 Purpose	31
29.2 Arguments	31
29.3 Return Value:	31
29.4 C Bindings	32
29.5 Fortran Bindings	32
29.6 Online Browsing	32
30 PLASMA_sgelqf_Tile	32
30.1 Purpose	32
30.2 Arguments	32
30.3 Return Value:	32
30.4 C Bindings	32
30.5 Fortran Bindings	32
30.6 Online Browsing	33

31 PLASMA_sgelqs	33
31.1 Purpose	33
31.2 Arguments	33
31.3 Return Value:	33
31.4 C Bindings	33
31.5 Fortran Bindings	33
31.6 Online Browsing	34
32 PLASMA_sgelqs_Tile	34
32.1 Purpose	34
32.2 Arguments	34
32.3 Return Value:	34
32.4 C Bindings	34
32.5 Fortran Bindings	34
32.6 Online Browsing	34
33 PLASMA_sgels	34
33.1 Purpose	34
33.2 Arguments	35
33.3 Return Value:	35
33.4 C Bindings	36
33.5 Fortran Bindings	36
33.6 Online Browsing	36
34 PLASMA_sgels_Tile	36
34.1 Purpose	36
34.2 Arguments	36
34.3 Return Value:	37
34.4 C Bindings	37
34.5 Fortran Bindings	37
34.6 Online Browsing	37
35 PLASMA_sgeqrf	37
35.1 Purpose	37
35.2 Arguments	37
35.3 Return Value:	38
35.4 C Bindings	38
35.5 Fortran Bindings	38
35.6 Online Browsing	38

36 PLASMA_sgeqrf_Tile	38
36.1 Purpose	38
36.2 Arguments	38
36.3 Return Value:	39
36.4 C Bindings	39
36.5 Fortran Bindings	39
36.6 Online Browsing	39
37 PLASMA_sgeqrs	39
37.1 Purpose	39
37.2 Arguments	39
37.3 Return Value:	40
37.4 C Bindings	40
37.5 Fortran Bindings	40
37.6 Online Browsing	40
38 PLASMA_sgeqrs_Tile	40
38.1 Purpose	40
38.2 Arguments	40
38.3 Return Value:	40
38.4 C Bindings	40
38.5 Fortran Bindings	41
38.6 Online Browsing	41
39 PLASMA_sgesv	41
39.1 Purpose	41
39.2 Arguments	41
39.3 Return Value:	41
39.4 C Bindings	42
39.5 Fortran Bindings	42
39.6 Online Browsing	42
40 PLASMA_sgesv_Tile	42
40.1 Purpose	42
40.2 Arguments	42
40.3 Return Value:	42
40.4 C Bindings	43
40.5 Fortran Bindings	43
40.6 Online Browsing	43

41 PLASMA_sgetrf	43
41.1 Purpose	43
41.2 Arguments	43
41.3 Return Value:	43
41.4 C Bindings	44
41.5 Fortran Bindings	44
41.6 Online Browsing	44
42 PLASMA_sgetrf_Tile	44
42.1 Purpose	44
42.2 Arguments	44
42.3 Return Value:	44
42.4 C Bindings	44
42.5 Fortran Bindings	44
42.6 Online Browsing	45
43 PLASMA_sgetrs	45
43.1 Purpose	45
43.2 Arguments	45
43.3 Return Value:	45
43.4 C Bindings	46
43.5 Fortran Bindings	46
43.6 Online Browsing	46
44 PLASMA_sgetrs_Tile	46
44.1 Purpose	46
44.2 Arguments	46
44.3 Return Value:	46
44.4 C Bindings	46
44.5 Fortran Bindings	47
44.6 Online Browsing	47
45 PLASMA_sposv	47
45.1 Purpose	47
45.2 Arguments	47
45.3 Return Value:	48
45.4 C Bindings	48
45.5 Fortran Bindings	48
45.6 Online Browsing	48

46 PLASMA_sposv_Tile	48
46.1 Purpose	48
46.2 Arguments	48
46.3 Return Value:	49
46.4 C Bindings	49
46.5 Fortran Bindings	49
46.6 Online Browsing	49
47 PLASMA_spotrf	49
47.1 Purpose	49
47.2 Arguments	50
47.3 Return Value:	50
47.4 C Bindings	50
47.5 Fortran Bindings	50
47.6 Online Browsing	50
48 PLASMA_spotrf_Tile	50
48.1 Purpose	50
48.2 Arguments	51
48.3 Return Value:	51
48.4 C Bindings	51
48.5 Fortran Bindings	51
48.6 Online Browsing	51
49 PLASMA_spotrs	51
49.1 Purpose	51
49.2 Arguments	52
49.3 Return Value:	52
49.4 C Bindings	52
49.5 Fortran Bindings	52
49.6 Online Browsing	52
50 PLASMA_spotrs_Tile	52
50.1 Purpose	52
50.2 Arguments	53
50.3 Return Value:	53
50.4 C Bindings	53
50.5 Fortran Bindings	53
50.6 Online Browsing	53

51 PLASMA_strsm	53
51.1 Purpose	53
51.2 Arguments	53
51.3 Return Value:	54
51.4 C Bindings	54
51.5 Fortran Bindings	54
51.6 Online Browsing	54
52 PLASMA_strsm_Tile	55
52.1 Purpose	55
52.2 Arguments	55
52.3 Return Value:	55
52.4 C Bindings	55
52.5 Fortran Bindings	56
52.6 Online Browsing	56
53 PLASMA_strsmpl	56
53.1 Purpose	56
53.2 Arguments	56
53.3 Return Value:	56
53.4 C Bindings	57
53.5 Fortran Bindings	57
53.6 Online Browsing	57
54 PLASMA_strsmpl_Tile	57
54.1 Purpose	57
54.2 Arguments	57
54.3 Return Value:	57
54.4 C Bindings	57
54.5 Fortran Bindings	57
54.6 Online Browsing	58
55 PLASMA_zcgesv	58
55.1 Purpose	58
55.2 Arguments	58
55.3 Return Value:	59
55.4 C Bindings	59
55.5 Fortran Bindings	59
55.6 Online Browsing	59

56 PLASMA_zcgesv_Tile	59
56.1 Purpose	59
56.2 Arguments	60
56.3 Return Value:	61
56.4 C Bindings	61
56.5 Fortran Bindings	61
56.6 Online Browsing	61
57 PLASMA_zgelqf	61
57.1 Purpose	61
57.2 Arguments	61
57.3 Return Value:	62
57.4 C Bindings	62
57.5 Fortran Bindings	62
57.6 Online Browsing	62
58 PLASMA_zgelqf_Tile	62
58.1 Purpose	62
58.2 Arguments	62
58.3 Return Value:	62
58.4 C Bindings	63
58.5 Fortran Bindings	63
58.6 Online Browsing	63
59 PLASMA_zgelqs	63
59.1 Purpose	63
59.2 Arguments	63
59.3 Return Value:	63
59.4 C Bindings	64
59.5 Fortran Bindings	64
59.6 Online Browsing	64
60 PLASMA_zgelqs_Tile	64
60.1 Purpose	64
60.2 Arguments	64
60.3 Return Value:	64
60.4 C Bindings	64
60.5 Fortran Bindings	64
60.6 Online Browsing	65

61 PLASMA_zgels	65
61.1 Purpose	65
61.2 Arguments	65
61.3 Return Value:	66
61.4 C Bindings	66
61.5 Fortran Bindings	66
61.6 Online Browsing	66
62 PLASMA_zgels_Tile	66
62.1 Purpose	66
62.2 Arguments	67
62.3 Return Value:	67
62.4 C Bindings	67
62.5 Fortran Bindings	67
62.6 Online Browsing	67
63 PLASMA_zgeqrf	67
63.1 Purpose	67
63.2 Arguments	68
63.3 Return Value:	68
63.4 C Bindings	68
63.5 Fortran Bindings	68
63.6 Online Browsing	68
64 PLASMA_zgeqrf_Tile	68
64.1 Purpose	68
64.2 Arguments	69
64.3 Return Value:	69
64.4 C Bindings	69
64.5 Fortran Bindings	69
64.6 Online Browsing	69
65 PLASMA_zgeqrs	69
65.1 Purpose	69
65.2 Arguments	69
65.3 Return Value:	70
65.4 C Bindings	70
65.5 Fortran Bindings	70
65.6 Online Browsing	70

66 PLASMA_zgeqrs_Tile	70
66.1 Purpose	70
66.2 Arguments	71
66.3 Return Value:	71
66.4 C Bindings	71
66.5 Fortran Bindings	71
66.6 Online Browsing	71
67 PLASMA_zgesv	71
67.1 Purpose	71
67.2 Arguments	71
67.3 Return Value:	72
67.4 C Bindings	72
67.5 Fortran Bindings	72
67.6 Online Browsing	72
68 PLASMA_zgesv_Tile	72
68.1 Purpose	72
68.2 Arguments	73
68.3 Return Value:	73
68.4 C Bindings	73
68.5 Fortran Bindings	73
68.6 Online Browsing	73
69 PLASMA_zgetrf	73
69.1 Purpose	73
69.2 Arguments	74
69.3 Return Value:	74
69.4 C Bindings	74
69.5 Fortran Bindings	74
69.6 Online Browsing	74
70 PLASMA_zgetrf_Tile	74
70.1 Purpose	74
70.2 Arguments	75
70.3 Return Value:	75
70.4 C Bindings	75
70.5 Fortran Bindings	75
70.6 Online Browsing	75

71 PLASMA_zgetrs	75
71.1 Purpose	75
71.2 Arguments	75
71.3 Return Value:	76
71.4 C Bindings	76
71.5 Fortran Bindings	76
71.6 Online Browsing	76
72 PLASMA_zgetrs_Tile	76
72.1 Purpose	76
72.2 Arguments	77
72.3 Return Value:	77
72.4 C Bindings	77
72.5 Fortran Bindings	77
72.6 Online Browsing	77
73 PLASMA_zposv	77
73.1 Purpose	77
73.2 Arguments	78
73.3 Return Value:	78
73.4 C Bindings	78
73.5 Fortran Bindings	78
73.6 Online Browsing	79
74 PLASMA_zposv_Tile	79
74.1 Purpose	79
74.2 Arguments	79
74.3 Return Value:	79
74.4 C Bindings	79
74.5 Fortran Bindings	80
74.6 Online Browsing	80
75 PLASMA_zpotrf	80
75.1 Purpose	80
75.2 Arguments	80
75.3 Return Value:	80
75.4 C Bindings	81
75.5 Fortran Bindings	81
75.6 Online Browsing	81

76 PLASMA_zpotrf_Tile	81
76.1 Purpose	81
76.2 Arguments	81
76.3 Return Value:	81
76.4 C Bindings	82
76.5 Fortran Bindings	82
76.6 Online Browsing	82
77 PLASMA_zpotrs	82
77.1 Purpose	82
77.2 Arguments	82
77.3 Return Value:	82
77.4 C Bindings	83
77.5 Fortran Bindings	83
77.6 Online Browsing	83
78 PLASMA_zpotrs_Tile	83
78.1 Purpose	83
78.2 Arguments	83
78.3 Return Value:	83
78.4 C Bindings	83
78.5 Fortran Bindings	83
78.6 Online Browsing	84
79 PLASMA_ztrsm	84
79.1 Purpose	84
79.2 Arguments	84
79.3 Return Value:	85
79.4 C Bindings	85
79.5 Fortran Bindings	85
79.6 Online Browsing	85
80 PLASMA_ztrsm_Tile	85
80.1 Purpose	85
80.2 Arguments	85
80.3 Return Value:	86
80.4 C Bindings	86
80.5 Fortran Bindings	86
80.6 Online Browsing	86

81 PLASMA_ztrsmpi	86
81.1 Purpose	86
81.2 Arguments	86
81.3 Return Value:	87
81.4 C Bindings	87
81.5 Fortran Bindings	87
81.6 Online Browsing	87
82 PLASMA_ztrsmpi_Tile	87
82.1 Purpose	87
82.2 Arguments	88
82.3 Return Value:	88
82.4 C Bindings	88
82.5 Fortran Bindings	88
82.6 Online Browsing	88
83 PLASMA_zunglq	88
83.1 Purpose	88
83.2 Arguments	88
83.3 Return Value:	89
83.4 C Bindings	89
83.5 Fortran Bindings	89
83.6 Online Browsing	89
84 PLASMA_zunglq_Tile	89
84.1 Purpose	89
84.2 Arguments	90
84.3 Return Value:	90
84.4 C Bindings	90
84.5 Fortran Bindings	90
84.6 Online Browsing	90
85 PLASMA_zungqr	90
85.1 Purpose	90
85.2 Arguments	90
85.3 Return Value:	91
85.4 C Bindings	91
85.5 Fortran Bindings	91
85.6 Online Browsing	91

86 PLASMA_zungqr_Tile	91
86.1 Purpose	91
86.2 Arguments	91
86.3 Return Value:	92
86.4 C Bindings	92
86.5 Fortran Bindings	92
86.6 Online Browsing	92
87 PLASMA_zunmlq	92
87.1 Purpose	92
87.2 Arguments	92
87.3 Return Value:	93
87.4 C Bindings	93
87.5 Fortran Bindings	93
87.6 Online Browsing	93
88 PLASMA_zunmlq_Tile	93
88.1 Purpose	93
88.2 Arguments	93
88.3 Return Value:	94
88.4 C Bindings	94
88.5 Fortran Bindings	94
88.6 Online Browsing	94
89 PLASMA_zunmqr	94
89.1 Purpose	94
89.2 Arguments	94
89.3 Return Value:	95
89.4 C Bindings	95
89.5 Fortran Bindings	95
89.6 Online Browsing	95
90 PLASMA_zunmqr_Tile	96
90.1 Purpose	96
90.2 Arguments	96
90.3 Return Value:	96
90.4 C Bindings	96
90.5 Fortran Bindings	96
90.6 Online Browsing	96

1 PLASMA_dgelqf

1.1 Purpose

PLASMA_dgelqf - Computes the tile LQ factorization of a complex M-by-N matrix A: $A = L * Q$.

1.2 Arguments

M	<code>int</code> (IN) The number of rows of the matrix A. $M \geq 0$.
N	<code>int</code> (IN) The number of columns of the matrix A. $N \geq 0$.
A	<code>double*</code> (INOUT) On entry, the M-by-N matrix A. On exit, the elements on and below the diagonal of the array contain the m-by-min(M,N) lower trapezoidal matrix L (L is lower triangular if $M \leq N$); the elements above the diagonal represent the unitary matrix Q as a product of elementary reflectors, stored by tiles.
LDA	<code>int</code> (IN) The leading dimension of the array A. $LDA \geq \max(1, M)$.
T	<code>double*</code> (OUT) On exit, auxiliary factorization data, required by PLASMA_dgelqs to solve the system of equations.

1.3 Return Value:

= 0: successful exit
< 0: if -i, the i-th argument had an illegal value

1.4 C Bindings

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

1.5 Fortran Bindings

```
void PLASMA_DGELQF(int *M, int *N, double *A, int *LDA, double **T, int *INFO)
```

1.6 Online Browsing

Dive into [PLASMA_dgelqf](#)

2 PLASMA_dgelqf_Tile

2.1 Purpose

PLASMA_dgelqf_Tile - Computes the tile LQ factorization of a complex M-by-N matrix A: $A = L * Q$. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

2.2 Arguments

A	<code>double* (INOUT)</code> On entry, the M-by-N matrix A. On exit, the elements on and below the diagonal of the array contain the m-by-min(\leftarrow M,N) lower trapezoidal matrix L (L is lower triangular <code>if</code> $M \leq N$); the elements above \leftarrow the diagonal represent the unitary matrix Q as a product of elementary reflectors, \leftarrow stored by tiles.
T	<code>double* (OUT)</code> On exit, auxiliary factorization data, required by PLASMA_dgelqs to solve the \leftarrow system of equations.

2.3 Return Value:

`= 0`: successful exit

2.4 C Bindings

```
int PLASMA_dgelqf_Tile(PLASMA_desc *A, PLASMA_desc *T)
```

2.5 Fortran Bindings

```
void PLASMA_DGELQF_TILE(long long int *A, long long int *T, int *INFO)
```

2.6 Online Browsing

Dive into [PLASMA_dgelqf_Tile](#)

3 PLASMA_dgelqs

3.1 Purpose

PLASMA_dgelqs - Compute a minimum-norm solution $\min \|A * X - B\|$ using the LQ factorization $A = L * Q$ computed by PLASMA_dgelqf.

3.2 Arguments

M	<code>int</code> (IN) The number of rows of the matrix A. $M \geq 0$.
N	<code>int</code> (IN) The number of columns of the matrix A. $N \geq M \geq 0$.
NRHS	<code>int</code> (IN) The number of columns of B. $NRHS \geq 0$.
A	<code>double*</code> (IN) Details of the LQ factorization of the original matrix A as returned by <code>PLASMA_dgelqf</code> .
LDA	<code>int</code> (IN) The leading dimension of the array A. $LDA \geq M$.
T	<code>double*</code> (IN) Auxiliary factorization data, computed by <code>PLASMA_dgelqf</code> .
B	<code>double*</code> (INOUT) On entry, the M-by-NRHS right hand side matrix B. On exit, the N-by-NRHS solution matrix X.
LDB	<code>int</code> (IN) The leading dimension of the array B. $LDB \geq N$.

3.3 Return Value:

= 0: successful exit
< 0: if -i, the i-th argument had an illegal value

3.4 C Bindings

```
int PLASMA_dgelqs(int M, int N, int NRHS, double *A, int LDA, double *T, double *B, int LDB ↵
)
```

3.5 Fortran Bindings

```
void PLASMA_DGELQS(int *M, int *N, int *NRHS, double *A, int *LDA, double **T, double *B, ↵
int *LDB, int *INFO)
```

3.6 Online Browsing

Dive into [PLASMA_dgelqs](#)

4 PLASMA_dgelqs_Tile

4.1 Purpose

PLASMA_dgelqs_Tile - Compute a minimum-norm solution $\min \|A^*X - B\|$ using the LQ factorization $A = L^*Q$ computed by `PLASMA_dgelqf_Tile`. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

4.2 Arguments

A	<code>double*</code> (IN) Details of the LQ factorization of the original matrix A as returned by <code>PLASMA_dgelqf</code> .
T	<code>double*</code> (IN) Auxiliary factorization data, computed by <code>PLASMA_dgelqf</code> .
B	<code>double*</code> (INOUT) On entry, the M-by-NRHS right hand side matrix B. On exit, the N-by-NRHS solution matrix X.

4.3 Return Value:

= 0: successful exit

4.4 C Bindings

```
int PLASMA_dgelqs_Tile(PLASMA_desc *A, PLASMA_desc *B, PLASMA_desc *T)
```

4.5 Fortran Bindings

```
void PLASMA_DGELQS_TILE(long long int *A, long long int *B, long long int *T, int *INFO)
```

4.6 Online Browsing

Dive into [PLASMA_dgelqs_Tile](#)

5 PLASMA_dgels

5.1 Purpose

PLASMA_dgels - solves overdetermined or underdetermined linear systems involving an M-by-N matrix A using the QR or the LQ factorization of A. It is assumed that A has full rank. The following options are provided:

1. `trans = PlasmaNoTrans` and $M \geq N$: find the least squares solution of an overdetermined system, i.e., solve the least squares problem: minimize $\|B - A * X\|$.
2. `trans = PlasmaNoTrans` and $M < N$: find the minimum norm solution of an underdetermined system $A * X = B$.

Several right hand side vectors B and solution vectors X can be handled in a single call; they are stored as the columns of the M-by-NRHS right hand side matrix B and the N-by-NRHS solution matrix X.

5.2 Arguments

trans	PLASMA_enum (IN) Intended usage: = PlasmaNoTrans: the linear system involves A; = PlasmaTrans: the linear system involves A**T. Currently only PlasmaNoTrans is supported.
M	int (IN) The number of rows of the matrix A. $M \geq 0$.
N	int (IN) The number of columns of the matrix A. $N \geq 0$.
NRHS	int (IN) The number of right hand sides, i.e., the number of columns of the matrices B and \leftarrow X. $NRHS \geq 0$.
A	double* (INOUT) On entry, the M-by-N matrix A. On exit, if $M \geq N$, A is overwritten by details of its QR factorization as returned by PLASMA_dgeqrf; if $M < N$, A is overwritten by details of its LQ factorization as returned by PLASMA_dgelqf.
LDA	int (IN) The leading dimension of the array A. $LDA \geq \max(1, M)$.
T	double* (OUT) On exit, auxiliary factorization data.
B	double* (INOUT) On entry, the M-by-NRHS matrix B of right hand side vectors, stored columnwise; On exit, if return value = 0, B is overwritten by the solution vectors, stored columnwise: if $M \geq N$, rows 1 to N of B contain the least squares solution vectors; the \leftarrow residual sum of squares for the solution in each column is given by the sum of squares of \leftarrow the modulus of elements N+1 to M in that column; if $M < N$, rows 1 to N of B contain the minimum norm solution vectors;
LDB	int (IN) The leading dimension of the array B. $LDB \geq \max(1, M, N)$.

5.3 Return Value:

= 0: successful exit
< 0: if -i, the i-th argument had an illegal value

5.4 C Bindings

```
int PLASMA_dgels(PLASMA_enum trans, int M, int N, int NRHS, double *A, int LDA, double *T,  $\leftarrow$ 
double *B, int LDB)
```

5.5 Fortran Bindings

```
void PLASMA_DGELS(PLASMA_enum *trans, int *M, int *N, int *NRHS, double *A, int *LDA,
double **T, double *B, int *LDB, int *INFO)
```

5.6 Online Browsing

Dive into [PLASMA_dgels](#)

6 PLASMA_dgels_Tile

6.1 Purpose

PLASMA_dgels_Tile - solves overdetermined or underdetermined linear systems involving an M-by-N matrix A using the QR or the LQ factorization of A. It is assumed that A has full rank. The following options are provided:

1. trans = PlasmaNoTrans and $M \geq N$: find the least squares solution of an overdetermined system, i.e., solve the least squares problem: minimize $\|B - A * X\|$.
2. trans = PlasmaNoTrans and $M < N$: find the minimum norm solution of an underdetermined system $A * X = B$.

Several right hand side vectors B and solution vectors X can be handled in a single call; they are stored as the columns of the M-by-NRHS right hand side matrix B and the N-by-NRHS solution matrix X. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

6.2 Arguments

trans	PLASMA_enum (IN) Intended usage: = PlasmaNoTrans: the linear system involves A; = PlasmaTrans: the linear system involves A**T. Currently only PlasmaNoTrans is supported.
A	double* (INOUT) On entry, the M-by-N matrix A. On exit, if $M \geq N$, A is overwritten by details of its QR factorization as returned by PLASMA_dgeqrf; if $M < N$, A is overwritten by details of its LQ factorization as returned by PLASMA_dgelqf.
T	double* (OUT) On exit, auxiliary factorization data.
B	double* (INOUT) On entry, the M-by-NRHS matrix B of right hand side vectors, stored columnwise; On exit, if return value = 0, B is overwritten by the solution vectors, stored columnwise: if $M \geq N$, rows 1 to N of B contain the least squares solution vectors; the residual sum of squares for the solution in each column is given by the sum of squares of the modulus of elements N+1 to M in that column; if $M < N$, rows 1 to N of B contain the minimum norm solution vectors;

6.3 Return Value:

```
= 0: successful exit
```

6.4 C Bindings

```
int PLASMA_dgels_Tile(PLASMA_enum trans, PLASMA_desc *A, PLASMA_desc *B, PLASMA_desc *T)
```

6.5 Fortran Bindings

```
void PLASMA_DGELS_TILE(PLASMA_enum *trans, long long int *A, long long int *B, long long  
int *T, int *INFO)
```

6.6 Online Browsing

Dive into [PLASMA_dgels_Tile](#)

7 PLASMA_dgeqrf

7.1 Purpose

PLASMA_dgeqrf - Computes the tile QR factorization of a complex M-by-N matrix A: $A = Q * R$.

7.2 Arguments

M	<code>int</code> (IN) The number of rows of the matrix A. $M \geq 0$.
N	<code>int</code> (IN) The number of columns of the matrix A. $N \geq 0$.
A	<code>double*</code> (INOUT) On entry, the M-by-N matrix A. On exit, the elements on and above the diagonal of the array contain the min(M,N)- by-N upper trapezoidal matrix R (R is upper triangular if $M \geq N$); the elements below the diagonal represent the unitary matrix Q as a product of elementary reflectors stored by tiles.
LDA	<code>int</code> (IN) The leading dimension of the array A. $LDA \geq \max(1, M)$.
T	<code>double*</code> (OUT) On exit, auxiliary factorization data, required by PLASMA_dgeqrs to solve the system of equations.

7.3 Return Value:

```
= 0: successful exit
< 0: if -i, the i-th argument had an illegal value
```

7.4 C Bindings

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

7.5 Fortran Bindings

```
void PLASMA_DGEQRF(int *M, int *N, double *A, int *LDA, double **T, int *INFO)
```

7.6 Online Browsing

Dive into [PLASMA_dgeqrf](#)

8 PLASMA_dgeqrf_Tile

8.1 Purpose

PLASMA_dgeqrf_Tile - Computes the tile QR factorization of a complex M-by-N matrix A: $A = Q * R$. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

8.2 Arguments

A	double* (INOUT) On entry, the M-by-N matrix A. On exit, the elements on and above the diagonal of the array contain the min(M,N)-by-N upper trapezoidal matrix R (R is upper triangular if $M \geq N$); the elements below the diagonal represent the unitary matrix Q as a product of elementary reflectors stored by tiles.
T	double* (OUT) On exit, auxiliary factorization data, required by PLASMA_dgeqrs to solve the system of equations.

8.3 Return Value:

```
= 0: successful exit
```

8.4 C Bindings

```
int PLASMA_dgeqrf_Tile(PLASMA_desc *A, PLASMA_desc *T)
```

8.5 Fortran Bindings

```
void PLASMA_DGEQRF_TILE(long long int *A, long long int *T, int *INFO)
```

8.6 Online Browsing

Dive into [PLASMA_dgeqrf_Tile](#)

9 PLASMA_dgeqrs

9.1 Purpose

PLASMA_dgeqrs - Compute a minimum-norm solution $\min \|A*X - B\|$ using the RQ factorization $A = R*Q$ computed by PLASMA_dgeqrf.

9.2 Arguments

M	<code>int</code> (IN) The number of rows of the matrix A. $M \geq 0$.
N	<code>int</code> (IN) The number of columns of the matrix A. $N \geq M \geq 0$.
NRHS	<code>int</code> (IN) The number of columns of B. $NRHS \geq 0$.
A	<code>double*</code> (INOUT) Details of the QR factorization of the original matrix A as returned by <code>PLASMA_dgeqrf</code> .
LDA	<code>int</code> (IN) The leading dimension of the array A. $LDA \geq M$.
T	<code>double*</code> (IN) Auxiliary factorization data, computed by PLASMA_dgeqrf.
B	<code>double*</code> (INOUT) On entry, the m-by-nrhs right hand side matrix B. On exit, the n-by-nrhs solution matrix X.
LDB	<code>int</code> (IN) The leading dimension of the array B. $LDB \geq \max(1, N)$.

9.3 Return Value:

```
= 0: successful exit
< 0: if -i, the i-th argument had an illegal value
```

9.4 C Bindings

```
int PLASMA_dgeqrs(int M, int N, int NRHS, double *A, int LDA, double *T, double *B, int LDB ↵
)
```

9.5 Fortran Bindings

```
void PLASMA_DGEQRS(int *M, int *N, int *NRHS, double *A, int *LDA, double **T, double *B,
                  int *LDB, int *INFO)
```

9.6 Online Browsing

Dive into [PLASMA_dgeqrs](#)

10 PLASMA_dgeqrs_Tile

10.1 Purpose

PLASMA_dgeqrs_Tile - Compute a minimum-norm solution $\min \|A^*X - B\|$ using the RQ factorization $A = R^*Q$ computed by PLASMA_dgeqrf_Tile. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

10.2 Arguments

A	<code>double*</code> (INOUT) Details of the QR factorization of the original matrix A as returned by PLASMA_dgeqrf.
T	<code>double*</code> (IN) Auxiliary factorization data, computed by PLASMA_dgeqrf.
B	<code>double*</code> (INOUT) On entry, the m-by-nrhs right hand side matrix B. On exit, the n-by-nrhs solution matrix X.

10.3 Return Value:

= 0: successful exit

10.4 C Bindings

```
int PLASMA_dgeqrs_Tile(PLASMA_desc *A, PLASMA_desc *B, PLASMA_desc *T)
```

10.5 Fortran Bindings

```
void PLASMA_DGEQRS_TILE(long long int *A, long long int *B, long long int *T, int *INFO)
```

10.6 Online Browsing

Dive into [PLASMA_dgeqrs_Tile](#)

11 PLASMA_dgesv

11.1 Purpose

PLASMA_dgesv - Computes the solution to a system of linear equations $A * X = B$, where A is an N -by- N matrix and X and B are N -by- $NRHS$ matrices. The tile LU decomposition with partial tile pivoting and row interchanges is used to factor A . The factored form of A is then used to solve the system of equations $A * X = B$.

11.2 Arguments

N	<code>int</code> (IN) The number of linear equations, i.e., the order of the matrix A . $N \geq 0$.
$NRHS$	<code>int</code> (IN) The number of right hand sides, i.e., the number of columns of the matrix B . $NRHS \geq 0$.
A	<code>double*</code> (INOUT) On entry, the N -by- N coefficient matrix A . On exit, the tile L and U factors from the factorization (not equivalent to LAPACK \leftrightarrow).
LDA	<code>int</code> (IN) The leading dimension of the array A . $LDA \geq \max(1, N)$.
L	<code>double*</code> (OUT) On exit, auxiliary factorization data, related to the tile L factor, necessary to solve the system of equations.
$IPIV$	<code>int*</code> (OUT) On exit, the pivot indices that define the permutations (not equivalent to LAPACK) \leftrightarrow .
B	<code>double*</code> (INOUT) On entry, the N -by- $NRHS$ matrix of right hand side matrix B . On exit, <code>if return</code> value = 0, the N -by- $NRHS$ solution matrix X .
LDB	<code>int</code> (IN) The leading dimension of the array B . $LDB \geq \max(1, N)$.

11.3 Return Value:

```

= 0: successful exit
< 0: if -i, the i-th argument had an illegal value
> 0: if i,  $U(i,i)$  is exactly zero. The factorization has been completed,
      but the factor  $U$  is exactly singular, so the solution could not be computed.

```

11.4 C Bindings

```

int PLASMA_dgesv(int N, int NRHS, double *A, int LDA, double *L, int *IPIV, double *B, int  $\leftrightarrow$ 
LDB)

```

11.5 Fortran Bindings

```
void PLASMA_DGESV(int *N, int *NRHS, double *A, int *LDA, double **LH, int **IPIVH, double *B, int *LDB, int *INFO) ↵
```

11.6 Online Browsing

Dive into [PLASMA_dgesv](#)

12 PLASMA_dgesv_Tile

12.1 Purpose

PLASMA_dgesv_Tile - Computes the solution to a system of linear equations $A * X = B$, where A is an N -by- N matrix and X and B are N -by- $NRHS$ matrices. The tile LU decomposition with partial tile pivoting and row interchanges is used to factor A . The factored form of A is then used to solve the system of equations $A * X = B$. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

12.2 Arguments

A	<code>double*</code> (INOUT) On entry, the N -by- N coefficient matrix A . On exit, the tile L and U factors from the factorization (not equivalent to LAPACK) ↵).
L	<code>double*</code> (OUT) On exit, auxiliary factorization data, related to the tile L factor, necessary to solve the system of equations.
IPIV	<code>int*</code> (OUT) On exit, the pivot indices that define the permutations (not equivalent to LAPACK) ↵ .
B	<code>double*</code> (INOUT) On entry, the N -by- $NRHS$ matrix of right hand side matrix B . On exit, if <code>return</code> value = 0, the N -by- $NRHS$ solution matrix X .

12.3 Return Value:

```
= 0: successful exit
> 0: if i, U(i,i) is exactly zero. The factorization has been completed,
    but the factor U is exactly singular, so the solution could not be computed.
```

12.4 C Bindings

```
int PLASMA_dgesv_Tile(PLASMA_desc *A, PLASMA_desc *L, int *IPIV, PLASMA_desc *B)
```

12.5 Fortran Bindings

```
void PLASMA_DGESV_TILE(long long int *A, long long int *L, int **IPIVH, long long int *B, ↵
int *INFO)
```

12.6 Online Browsing

Dive into [PLASMA_dgesv_Tile](#)

13 PLASMA_dgetrf

13.1 Purpose

PLASMA_dgetrf - Computes an LU factorization of a general M-by-N matrix A using the tile LU algorithm with partial tile pivoting with row interchanges.

13.2 Arguments

M	<code>int</code> (IN) The number of rows of the matrix A. $M \geq 0$.
N	<code>int</code> (IN) The number of columns of the matrix A. $N \geq 0$.
A	<code>double*</code> (INOUT) On entry, the M-by-N matrix to be factored. On exit, the tile factors L and U from the factorization.
LDA	<code>int</code> (IN) The leading dimension of the array A. $LDA \geq \max(1, M)$.
L	<code>double*</code> (OUT) On exit, auxiliary factorization data, related to the tile L factor, required by PLASMA_dgetrs to solve the system of equations.
IPIV	<code>int*</code> (OUT) The pivot indices that define the permutations (not equivalent to LAPACK).

13.3 Return Value:

```
= 0: successful exit
< 0: if -i, the i-th argument had an illegal value
> 0: if i, U(i,i) is exactly zero. The factorization has been completed,
      but the factor U is exactly singular, and division by zero will occur
      if it is used to solve a system of equations.
```

13.4 C Bindings

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

13.5 Fortran Bindings

```
void PLASMA_DGETRF(int *M, int *N, double *A, int *LDA, double **LH, int **IPIVH, int *INFO ←
)
```

13.6 Online Browsing

Dive into [PLASMA_dgetrf](#)

14 PLASMA_dgetrf_Tile

14.1 Purpose

PLASMA_dgetrf_Tile - Computes an LU factorization of a general M-by-N matrix A using the tile LU algorithm with partial tile pivoting with row interchanges. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

14.2 Arguments

A	<code>double*</code> (INOUT) On entry, the M-by-N matrix to be factored. On exit, the tile factors L and U from the factorization.
L	<code>double*</code> (OUT) On exit, auxiliary factorization data, related to the tile L factor, required by PLASMA_dgetrs to solve the system of equations.
IPIV	<code>int*</code> (OUT) The pivot indices that define the permutations (not equivalent to LAPACK).

14.3 Return Value:

```
= 0: successful exit
> 0: if i, U(i,i) is exactly zero. The factorization has been completed,
    but the factor U is exactly singular, and division by zero will occur
    if it is used to solve a system of equations.
```

14.4 C Bindings

```
int PLASMA_dgetrf_Tile(PLASMA_desc *A, PLASMA_desc *L, int *IPIV)
```

14.5 Fortran Bindings

```
void PLASMA_DGETRF_TILE(long long int *A, long long int *L, int **IPIVH, int *INFO)
```

14.6 Online Browsing

Dive into [PLASMA_dgetrf_Tile](#)

15 PLASMA_dgetrs

15.1 Purpose

PLASMA_dgetrs - Solves a system of linear equations $A * X = B$, with a general N-by-N matrix A using the tile LU factorization computed by PLASMA_dgetrf.

15.2 Arguments

trans	PLASMA_enum (IN) Intended to specify the the form of the system of equations: = PlasmaNoTrans: $A * X = B$ (No transpose) = PlasmaTrans: $A^{**T} * X = B$ (Transpose) = PlasmaTrans: $A^{**T} * X = B$ (Conjugate transpose) Currently only PlasmaNoTrans is supported.
N	int (IN) The order of the matrix A. $N \geq 0$.
NRHS	int (IN) The number of right hand sides, i.e., the number of columns of the matrix B. $NRHS \geq 0$.
A	double* (IN) The tile factors L and U from the factorization, computed by PLASMA_dgetrf.
LDA	int (IN) The leading dimension of the array A. $LDA \geq \max(1, N)$.
L	double* (IN) Auxiliary factorization data, related to the tile L factor, computed by \leftrightarrow PLASMA_dgetrf.
IPIV	int* (IN) The pivot indices from PLASMA_dgetrf (not equivalent to LAPACK).
B	double* (INOUT) On entry, the N-by-NRHS matrix of right hand side matrix B. On exit, the solution matrix X.
LDB	int (IN) The leading dimension of the array B. $LDB \geq \max(1, N)$.

15.3 Return Value:

= 0: successful exit
< 0: if -i, the i-th argument had an illegal value

15.4 C Bindings

```
int PLASMA_dgetrs(PLASMA_enum uplo, int N, int NRHS, double *A, int LDA, double *L, int *  $\leftrightarrow$ 
IPIV, double *B, int LDB)
```

15.5 Fortran Bindings

```
void PLASMA_DGETRS(PLASMA_enum *uplo, int *N, int *NRHS, double *A, int *LDA, double **LH,  $\leftrightarrow$ 
int **IPIVH, double *B, int *LDB, int *INFO)
```

15.6 Online Browsing

Dive into [PLASMA_dgetrs](#)

16 PLASMA_dgetrs_Tile

16.1 Purpose

PLASMA_dgetrs_Tile - Solves a system of linear equations $A * X = B$, with a general N-by-N matrix A using the tile LU factorization computed by PLASMA_dgetrf. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

16.2 Arguments

A	<code>double*</code> (IN) The tile factors L and U from the factorization, computed by PLASMA_dgetrf.
L	<code>double*</code> (IN) Auxiliary factorization data, related to the tile L factor, computed by \leftrightarrow PLASMA_dgetrf.
IPIV	<code>int*</code> (IN) The pivot indices from PLASMA_dgetrf (not equivalent to LAPACK).
B	<code>double*</code> (INOUT) On entry, the N-by-NRHS matrix of right hand side matrix B. On exit, the solution matrix X.

16.3 Return Value:

= 0: successful exit

16.4 C Bindings

```
int PLASMA_dgetrs_Tile(PLASMA_desc *A, PLASMA_desc *L, int *IPIV, PLASMA_desc *B)
```

16.5 Fortran Bindings

```
void PLASMA_DGETRS_TILE(long long int *A, long long int *L, int **IPIVH, long long int *B,  $\leftrightarrow$ 
int *INFO)
```

16.6 Online Browsing

Dive into [PLASMA_dgetrs_Tile](#)

17 PLASMA_dposv

17.1 Purpose

PLASMA_dposv - Computes the solution to a system of linear equations $A * X = B$, where A is an N-by-N symmetric positive definite (or Hermitian positive definite in the complex case) matrix and X and B are N-by-NRHS matrices. The Cholesky decomposition is used to factor A as

$A = U^{**T} * U$, if `uplo = PlasmaUpper`, or
 $A = L * L^{**T}$, if `uplo = PlasmaLower`,

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 $A * X = B$.

17.2 Arguments

<code>uplo</code>	<code>PLASMA_enum</code> (IN) Specifies whether the matrix A is upper triangular or lower triangular: = <code>PlasmaUpper</code> : Upper triangle of A is stored; = <code>PlasmaLower</code> : Lower triangle of A is stored.
<code>N</code>	<code>int</code> (IN) The number of linear equations, i.e., the order of the matrix A . $N \geq 0$.
<code>NRHS</code>	<code>int</code> (IN) The number of right hand sides, i.e., the number of columns of the matrix B . <code>NRHS</code> \leftarrow ≥ 0 .
<code>A</code>	<code>double*</code> (INOUT) On entry, the symmetric positive definite (or Hermitian) matrix A . If <code>uplo = PlasmaUpper</code> , the leading N -by- N upper triangular part of A contains the upper triangular part of the matrix A , and the strictly lower triangular part of A is not referenced. \leftarrow If <code>UPLO = 'L'</code> , the leading N -by- N lower triangular part of A contains the lower triangular part of the matrix A , and the strictly upper triangular part of A is not referenced. \leftarrow On exit, if <code>return</code> value = 0, the factor U or L from the Cholesky factorization $A = U^{**T} * U$ or $A = L * L^{**T}$.
<code>LDA</code>	<code>int</code> (IN) The leading dimension of the array A . <code>LDA</code> $\geq \max(1, N)$.
<code>B</code>	<code>double*</code> (INOUT) On entry, the N -by- <code>NRHS</code> right hand side matrix B . On exit, if <code>return</code> value = 0, the N -by- <code>NRHS</code> solution matrix X .
<code>LDB</code>	<code>int</code> (IN) The leading dimension of the array B . <code>LDB</code> $\geq \max(1, N)$.

17.3 Return Value:

= 0: successful exit
 < 0: if $-i$, the i -th argument had an illegal value
 > 0: if i , the leading minor of order i of A is not positive definite, so the factorization could not be completed, and the solution has not been computed \leftarrow
 .

17.4 C Bindings

```
int PLASMA_dposv(PLASMA_enum uplo, int N, int NRHS, double *A, int LDA, double *B, int LDB)
```

17.5 Fortran Bindings

```
void PLASMA_DPOSV(PLASMA_enum *uplo, int *N, int *NRHS, double *A, int *LDA, double *B, int *LDB, int *INFO)
```

17.6 Online Browsing

Dive into [PLASMA_dposv](#)

18 PLASMA_dposv_Tile

18.1 Purpose

PLASMA_dposv_Tile - Computes the solution to a system of linear equations $A * X = B$, where A is an N -by- N symmetric positive definite (or Hermitian positive definite in the complex case) matrix and X and B are N -by- $NRHS$ matrices. The Cholesky decomposition is used to factor A as

$A = U^{*}T * U$, if $uplo = PlasmaUpper$, or
 $A = L * L^{*}T$, if $uplo = PlasmaLower$,

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 $A * X = B$. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

18.2 Arguments

uplo	PLASMA_enum (IN) Specifies whether the matrix A is upper triangular or lower triangular: = PlasmaUpper: Upper triangle of A is stored; = PlasmaLower: Lower triangle of A is stored.
A	double* (INOUT) On entry, the symmetric positive definite (or Hermitian) matrix A . If $uplo = PlasmaUpper$, the leading N -by- N upper triangular part of A contains the upper triangular part of the matrix A , and the strictly lower triangular part of A is not referenced. If $UPLO = 'L'$, the leading N -by- N lower triangular part of A contains the lower triangular part of the matrix A , and the strictly upper triangular part of A is not referenced. On exit, if return value = 0, the factor U or L from the Cholesky factorization $A = U^{*}T*U$ or $A = L*L^{*}T$.
B	double* (INOUT) On entry, the N -by- $NRHS$ right hand side matrix B . On exit, if return value = 0, the N -by- $NRHS$ solution matrix X .

18.3 Return Value:

= 0: successful exit
 > 0: if i , the leading minor of order i of A is not positive definite, so the factorization could not be completed, and the solution has not been computed
 .

18.4 C Bindings

```
int PLASMA_dposv_Tile(PLASMA_enum uplo, PLASMA_desc *A, PLASMA_desc *B)
```

18.5 Fortran Bindings

```
void PLASMA_DPOSV_TILE(PLASMA_enum *uplo, long long int *A, long long int *B, int *INFO)
```

18.6 Online Browsing

Dive into [PLASMA_dposv_Tile](#)

19 PLASMA_dpotrf

19.1 Purpose

PLASMA_dpotrf - Computes the Cholesky factorization of a symmetric positive definite (or Hermitian positive definite in the complex case) matrix A. The factorization has the form

$A = U^*T * U$, if uplo = PlasmaUpper, or
 $A = L * L^*T$, if uplo = PlasmaLower,

where U is an upper triangular matrix and L is a lower triangular matrix.

19.2 Arguments

uplo	PLASMA_enum (IN) = PlasmaUpper: Upper triangle of A is stored; = PlasmaLower: Lower triangle of A is stored.
N	int (IN) The order of the matrix A. $N \geq 0$.
A	double* (INOUT) On entry, the symmetric positive definite (or Hermitian) matrix A. If uplo = PlasmaUpper, the leading N-by-N upper triangular part of A contains the upper triangular part of the matrix A, and the strictly lower triangular part of A is not referenced. If UPLO = 'L', the leading N-by-N lower triangular part of A contains the lower triangular part of the matrix A, and the strictly upper triangular part of A is not referenced. On exit, if return value = 0, the factor U or L from the Cholesky factorization $A = U^*T*U$ or $A = L*L^*T$.
LDA	int (IN) The leading dimension of the array A. $LDA \geq \max(1, N)$.

19.3 Return Value:

```

= 0: successful exit
< 0: if -i, the i-th argument had an illegal value
> 0: if i, the leading minor of order i of A is not positive definite, so the
      factorization could not be completed, and the solution has not been computed ←
      .

```

19.4 C Bindings

```
int PLASMA_dpotrf(PLASMA_enum uplo, int N, double *A, int LDA)
```

19.5 Fortran Bindings

```
void PLASMA_DPOTRF(PLASMA_enum *uplo, int *N, double *A, int *LDA, int *INFO)
```

19.6 Online Browsing

Dive into [PLASMA_dpotrf](#)

20 PLASMA_dpotrf_Tile

20.1 Purpose

PLASMA_dpotrf_Tile - Computes the Cholesky factorization of a symmetric positive definite (or Hermitian positive definite in the complex case) matrix A. The factorization has the form

$A = U^{*}T * U$, if uplo = PlasmaUpper, or
 $A = L * L^{*}T$, if uplo = PlasmaLower,

where U is an upper triangular matrix and L is a lower triangular matrix. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

20.2 Arguments

uplo	PLASMA_enum (IN) = PlasmaUpper: Upper triangle of A is stored; = PlasmaLower: Lower triangle of A is stored.
A	double* (INOUT) On entry, the symmetric positive definite (or Hermitian) matrix A. If uplo = PlasmaUpper, the leading N-by-N upper triangular part of A contains the upper triangular part of the matrix A, and the strictly lower ← triangular part of A is not referenced. If UPLO = 'L', the leading N-by-N lower triangular part of A contains the lower triangular part of the matrix A, and the strictly upper triangular part of A is ← not referenced. On exit, if return value = 0, the factor U or L from the Cholesky factorization $A = U^{*}T * U$ or $A = L * L^{*}T$.

20.3 Return Value:

```

= 0: successful exit
> 0: if i, the leading minor of order i of A is not positive definite, so the
      factorization could not be completed, and the solution has not been computed ↵
      .

```

20.4 C Bindings

```
int PLASMA_dpotrf_Tile(PLASMA_enum uplo, PLASMA_desc *A)
```

20.5 Fortran Bindings

```
void PLASMA_DPOTRF_TILE(PLASMA_enum *uplo, long long int *A, int *INFO)
```

20.6 Online Browsing

Dive into [PLASMA_dpotrf_Tile](#)

21 PLASMA_dpotrs

21.1 Purpose

PLASMA_dpotrs - Solves a system of linear equations $A * X = B$ with a symmetric positive definite (or Hermitian positive definite in the complex case) matrix A using the Cholesky factorization $A = UT^*U$ or $A = L^*LT$ computed by **PLASMA_dpotrf**.

21.2 Arguments

uplo	PLASMA_enum (IN) = PlasmaUpper: Upper triangle of A is stored; = PlasmaLower: Lower triangle of A is stored.
N	int (IN) The order of the matrix A. $N \geq 0$.
NRHS	int (IN) The number of right hand sides, i.e., the number of columns of the matrix B. NRHS ↵ ≥ 0 .
A	double* (IN) The triangular factor U or L from the Cholesky factorization $A = U^*T^*U$ or $A = L^*L$ ↵ *T , computed by PLASMA_dpotrf .
LDA	int (IN) The leading dimension of the array A. $LDA \geq \max(1, N)$.
B	double* (INOUT) On entry, the N-by-NRHS right hand side matrix B. On exit, if return value = 0, the N-by-NRHS solution matrix X.
LDB	int (IN) The leading dimension of the array B. $LDB \geq \max(1, N)$.

21.3 Return Value:

```
= 0: successful exit
< 0: if -i, the i-th argument had an illegal value
```

21.4 C Bindings

```
int PLASMA_dpotrs(PLASMA_enum uplo, int N, int NRHS, double *A, int LDA, double *B, int LDB ↵
)
```

21.5 Fortran Bindings

```
void PLASMA_DPOTRS(PLASMA_enum *uplo, int *N, int *NRHS, double *A, int *LDA, double *B, ↵
int* LDB, int * INFO)
```

21.6 Online Browsing

Dive into [PLASMA_dpotrs](#)

22 PLASMA_dpotrs_Tile

22.1 Purpose

PLASMA_dpotrs_Tile - Solves a system of linear equations $A * X = B$ with a symmetric positive definite (or Hermitian positive definite in the complex case) matrix A using the Cholesky factorization $A = UT*U$ or $A = L*LT$ computed by `PLASMA_dpotrf`. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

22.2 Arguments

uplo	PLASMA_enum (IN) = PlasmaUpper: Upper triangle of A is stored; = PlasmaLower: Lower triangle of A is stored.
A	double* (IN) The triangular factor U or L from the Cholesky factorization $A = U**T*U$ or $A = L*L ↵$ **T, computed by <code>PLASMA_dpotrf</code> .
B	double* (INOUT) On entry, the N-by-NRHS right hand side matrix B. On exit, if return value = 0, the N-by-NRHS solution matrix X.

22.3 Return Value:

```
= 0: successful exit
```

22.4 C Bindings

```
int PLASMA_dpotrs_Tile(PLASMA_enum uplo, PLASMA_desc *A, PLASMA_desc *B)
```

22.5 Fortran Bindings

```
void PLASMA_DPOTRS_TILE(PLASMA_enum *uplo, long long int *A, long long int *B, int *INFO)
```

22.6 Online Browsing

Dive into [PLASMA_dpotrs_Tile](#)

23 PLASMA_dsgesv

23.1 Purpose

PLASMA_dsgesv - Computes the solution to a system of linear equations $A * X = B$, where A is an N -by- N matrix and X and B are N -by- $NRHS$ matrices. The tile LU decomposition with partial tile pivoting and row interchanges is used to factor A . The factored form of A is then used to solve the system of equations $A * X = B$.

IMPORTANT NOTICE: in its current state, this routine only intends to be a proof-of-concept. There are still some costly serial parts and one may NOT expect to achieve high performance.

PLASMA_dsgesv first attempts to factorize the matrix in COMPLEX and use this factorization within an iterative refinement procedure to produce a solution with COMPLEX*16 normwise backward error quality (see below). If the approach fails the method switches to a COMPLEX*16 factorization and solve.

The iterative refinement is not going to be a winning strategy if the ratio COMPLEX performance over COMPLEX*16 performance is too small. A reasonable strategy should take the number of right-hand sides and the size of the matrix into account. This might be done with a call to ILAENV in the future. Up to now, we always try iterative refinement.

The iterative refinement process is stopped if $ITER > ITERMAX$ or for all the RHS we have: $RNRM < SQRT(N) * XNRM * ANRM * EPS$ where

- $ITER$ is the number of the current iteration in the iterative refinement process
- $RNRM$ is the infinity-norm of the residual
- $XNRM$ is the infinity-norm of the solution
- $ANRM$ is the infinity-operator-norm of the matrix A
- EPS is the machine epsilon returned by `DLAMCH(Epsilon)`.

The values $ITERMAX$ and $BWDMAX$ are fixed to 30 and 1.0D+00 respectively.

23.2 Arguments

N	<code>int</code> (IN) The number of linear equations, i.e., the order of the matrix A . $N \geq 0$.
$NRHS$	<code>int</code> (IN) The number of right hand sides, i.e., the number of columns of the matrix B . $NRHS \geq 0$.

```

A      double* (INOUT)
      On entry, the N-by-N coefficient matrix A.
      On exit, the tile L and U factors from the factorization (not equivalent to LAPACK ←
      ).

LDA    int (IN)
      The leading dimension of the array A. LDA >= max(1,N).

L      double* (OUT)
      On exit, auxiliary factorization data, related to the tile L factor,
      necessary to solve the system of equations.

IPIV   int* (OUT)
      On exit, the pivot indices that define the permutations (not equivalent to LAPACK) ←
      .

B      double* (IN)
      The N-by-NRHS matrix of right hand side matrix B.

LDB    int (IN)
      The leading dimension of the array B. LDB >= max(1,N).

X      double* (OUT)
      If return value = 0, the N-by-NRHS solution matrix X.

LDX    int (IN)
      The leading dimension of the array B. LDX >= max(1,N).

ITER   int* (OUT) is the number of the current iteration in the iterative refinement ←
      process

```

23.3 Return Value:

```

= 0: successful exit
< 0: if -i, the i-th argument had an illegal value
> 0: if i, U(i,i) is exactly zero. The factorization has been completed,
      but the factor U is exactly singular, so the solution could not be computed.

```

23.4 C Bindings

```

int PLASMA_dsgesv(int N, int NRHS, double *A, int LDA, double *L, int *IPIV, double *B, int ←
    LDB, double *X, int LDX, int *ITER)

```

23.5 Fortran Bindings

```

void PLASMA_DSGETV(int *N, int *NRHS, double *A, int *LDA, double **LH, int **IPIVH, double ←
    *B, int *LDB, double *X, int *LDX, int *ITER, int *INFO)

```

23.6 Online Browsing

Dive into [PLASMA_dsgesv](#)

24 PLASMA_dsgesv_Tile

24.1 Purpose

PLASMA_dsgesv - Computes the solution to a system of linear equations $A * X = B$, where A is an N -by- N matrix and X and B are N -by- $NRHS$ matrices. The tile LU decomposition with partial tile pivoting and row interchanges is used to factor A . The factored form of A is then used to solve the system of equations $A * X = B$. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

IMPORTANT NOTICE: in its current state, this routine only intends to be a proof-of-concept. There are still some costly serial parts and one may NOT expect to achieve high performance.

PLASMA_dsgesv first attempts to factorize the matrix in COMPLEX and use this factorization within an iterative refinement procedure to produce a solution with COMPLEX*16 normwise backward error quality (see below). If the approach fails the method switches to a COMPLEX*16 factorization and solve.

The iterative refinement is not going to be a winning strategy if the ratio COMPLEX performance over COMPLEX*16 performance is too small. A reasonable strategy should take the number of right-hand sides and the size of the matrix into account. This might be done with a call to ILAENV in the future. Up to now, we always try iterative refinement.

The iterative refinement process is stopped if $ITER > ITERMAX$ or for all the RHS we have: $RNRM < SQRT(N) * XNRM * ANRM * EPS$ where

- $ITER$ is the number of the current iteration in the iterative refinement process
- $RNRM$ is the infinity-norm of the residual
- $XNRM$ is the infinity-norm of the solution
- $ANRM$ is the infinity-operator-norm of the matrix A
- EPS is the machine epsilon returned by `DLAMCH(Epsilon)`.

The values $ITERMAX$ and $BWDMAX$ are fixed to 30 and 1.0D+00 respectively.

24.2 Arguments

N	<code>int</code> (IN) The number of linear equations, i.e., the order of the matrix A . $N \geq 0$.
$NRHS$	<code>int</code> (IN) The number of right hand sides, i.e., the number of columns of the matrix B . $NRHS \geq 0$.
A	<code>double*</code> (INOUT) On entry, the N -by- N coefficient matrix A . On exit, the tile L and U factors from the factorization (not equivalent to LAPACK \leftarrow).
LDA	<code>int</code> (IN) The leading dimension of the array A . $LDA \geq \max(1, N)$.
L	<code>double*</code> (OUT) On exit, auxiliary factorization data, related to the tile L factor, necessary to solve the system of equations.
$IPIV$	<code>int*</code> (OUT) On exit, the pivot indices that define the permutations (not equivalent to LAPACK \leftarrow).

B `double*` (IN)
The N-by-NRHS matrix of right hand side matrix B.

LDB `int` (IN)
The leading dimension of the array B. `LDB >= max(1,N)`.

X `double*` (OUT)
If `return` value = 0, the N-by-NRHS solution matrix X.

LDX `int` (IN)
The leading dimension of the array B. `LDX >= max(1,N)`.

ITER `int*` (OUT) is the number of the current iteration in the iterative refinement process \leftrightarrow

24.3 Return Value:

```
= 0: successful exit
< 0: if -i, the i-th argument had an illegal value
> 0: if i, U(i,i) is exactly zero. The factorization has been completed,
      but the factor U is exactly singular, so the solution could not be computed.
```

24.4 C Bindings

24.5 Fortran Bindings

```
void PLASMA_DSGESV_TILE(long long int *A, long long int *L, int **IPIVH, long long int *B,  $\leftrightarrow$ 
    long long *X, int *ITER, int *INFO)
```

24.6 Online Browsing

Dive into [PLASMA_dsgesv_Tile](#)

25 PLASMA_dtrsm

25.1 Purpose

PLASMA_dtrsm - Computes triangular solve $A*X = B$ or $X*A = B$

25.2 Arguments

side `PLASMA_enum` (IN)
Specifies whether A appears on the left or on the right of X:
= `PlasmaLeft`: $A*X = B$
= `PlasmaRight`: $X*A = B$

uplo `PLASMA_enum` (IN)
Specifies whether the matrix A is upper triangular or lower triangular:
= `PlasmaUpper`: Upper triangle of A is stored;

```

        = PlasmaLower: Lower triangle of A is stored.

transA    PLASMA_enum (IN)
    Specifies whether the matrix A is transposed, not transposed or conjugate ←
        transposed:
        = PlasmaNoTrans:  A is transposed;
        = PlasmaTrans:   A is not transposed;
        = PlasmaTrans: A is conjugate transposed.

diag      PLASMA_enum (IN)
    Specifies whether or not A is unit triangular:
    = PlasmaNonUnit: A is non unit;
    = PlasmaUnit:   A is unit.

N         int (IN)
    The order of the matrix A. N >= 0.

NRHS      int (IN)
    The number of right hand sides, i.e., the number of columns of the matrix B. NRHS ←
    >= 0.

A         double* (IN)
    The triangular matrix A. If uplo = PlasmaUpper, the leading N-by-N upper ←
    triangular
    part of the array A contains the upper triangular matrix, and the strictly lower
    triangular part of A is not referenced. If uplo = PlasmaLower, the leading N-by-N
    lower triangular part of the array A contains the lower triangular matrix, and the
    strictly upper triangular part of A is not referenced. If diag = PlasmaUnit, the
    diagonal elements of A are also not referenced and are assumed to be 1.

LDA       int (IN)
    The leading dimension of the array A. LDA >= max(1,N).

B         double* (INOUT)
    On entry, the N-by-NRHS right hand side matrix B.
    On exit, if return value = 0, the N-by-NRHS solution matrix X.

LDB       double* (IN)
    The leading dimension of the array B. LDB >= max(1,N).

```

25.3 Return Value:

```

    = 0: successful exit
    < 0: if -i, the i-th argument had an illegal value

```

25.4 C Bindings

```

int PLASMA_dtrsm(PLASMA_enum side, PLASMA_enum uplo, PLASMA_enum transA, PLASMA_enum diag, ←
    int N, int NRHS, double *A, int LDA, double *B, int LDB)

```

25.5 Fortran Bindings

```

void PLASMA_DTRSM(PLASMA_enum *side, PLASMA_enum *uplo, PLASMA_enum *transA, PLASMA_enum * ←
    diag, int *N, int *NRHS, double *A, int *LDA, double *B, int *LDB, int *INFO)

```

25.6 Online Browsing

Dive into [PLASMA_dtrsm](#)

26 PLASMA_dtrsm_Tile

26.1 Purpose

PLASMA_dtrsm_Tile - Computes triangular solve $A*X = B$ or $X*A = B$ All matrices are passed through descriptors. All dimensions are taken from the descriptors.

26.2 Arguments

side	PLASMA_enum (IN) Specifies whether A appears on the left or on the right of X: = PlasmaLeft: $A*X = B$ = PlasmaRight: $X*A = B$
uplo	PLASMA_enum (IN) Specifies whether the matrix A is upper triangular or lower triangular: = PlasmaUpper: Upper triangle of A is stored; = PlasmaLower: Lower triangle of A is stored.
transA	PLASMA_enum (IN) Specifies whether the matrix A is transposed, not transposed or conjugate transposed: \leftrightarrow = PlasmaNoTrans: A is transposed; = PlasmaTrans: A is not transposed; = PlasmaTrans: A is conjugate transposed.
diag	PLASMA_enum (IN) Specifies whether or not A is unit triangular: = PlasmaNonUnit: A is non unit; = PlasmaUnit: A is unit.
A	double* (IN) The triangular matrix A. If uplo = PlasmaUpper, the leading N-by-N upper triangular part of the array A contains the upper triangular matrix, and the strictly lower triangular part of A is not referenced. If uplo = PlasmaLower, the leading N-by-N lower triangular part of the array A contains the lower triangular matrix, and the strictly upper triangular part of A is not referenced. If diag = PlasmaUnit, the diagonal elements of A are also not referenced and are assumed to be 1. \leftrightarrow
B	double* (INOUT) On entry, the N-by-NRHS right hand side matrix B. On exit, if return value = 0, the N-by-NRHS solution matrix X.

26.3 Return Value:

= 0: successful exit

26.4 C Bindings

```
int PLASMA_dtrsm_Tile(PLASMA_enum side, PLASMA_enum uplo, PLASMA_enum transA, PLASMA_enum ↵
    diag, PLASMA_desc *A, PLASMA_desc *B)
```

26.5 Fortran Bindings

```
void PLASMA_DTRSM_TILE(PLASMA_enum *side, PLASMA_enum *uplo, PLASMA_enum *transA, ↵
    PLASMA_enum *diag, long long int *A, long long int *B, int *INFO)
```

26.6 Online Browsing

Dive into [PLASMA_dtrsm_Tile](#)

27 PLASMA_dtrsmpi

27.1 Purpose

PLASMA_dtrsmpi - Performs the forward substitution step of solving a system of linear equations after the tile LU factorization of the matrix.

27.2 Arguments

N	<code>int</code> (IN) The order of the matrix A. $N \geq 0$.
NRHS	<code>int</code> (IN) The number of right hand sides, i.e., the number of columns of the matrix B. $NRHS \geq 0$. ↵
A	<code>double*</code> (IN) The tile factor L from the factorization, computed by <code>PLASMA_dgetrf</code> .
LDA	<code>int</code> (IN) The leading dimension of the array A. $LDA \geq \max(1, N)$.
L	<code>double*</code> (IN) Auxiliary factorization data, related to the tile L factor, computed by <code>PLASMA_dgetrf</code> . ↵
IPIV	<code>int*</code> (IN) The pivot indices from <code>PLASMA_dgetrf</code> (not equivalent to LAPACK).
B	<code>double*</code> (INOUT) On entry, the N-by-NRHS right hand side matrix B. On exit, <code>if return</code> value = 0, the N-by-NRHS solution matrix X.
LDB	<code>double*</code> (IN) The leading dimension of the array B. $LDB \geq \max(1, N)$.

27.3 Return Value:

```
= 0: successful exit
< 0: if -i, the i-th argument had an illegal value
```

27.4 C Bindings

```
int PLASMA_dtrsmp1(int N, int NRHS, double *A, int LDA, double *L, int *IPIV, double *B, ←
int LDB)
```

27.5 Fortran Bindings

```
void PLASMA_DTRSMP1(int *N, int *NRHS, double *A, int *LDA, double **LH, int **IPIVH, ←
double *B, int *LDB, int *INFO)
```

27.6 Online Browsing

Dive into [PLASMA_dtrsmp1](#)

28 PLASMA_dtrsmp1_Tile

28.1 Purpose

PLASMA_dtrsmp1_Tile - Performs the forward substitution step of solving a system of linear equations after the tile LU factorization of the matrix. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

28.2 Arguments

A	<code>double*</code> (IN) The tile factor L from the factorization, computed by <code>PLASMA_dgetrf</code> .
L	<code>double*</code> (IN) Auxiliary factorization data, related to the tile L factor, computed by ← <code>PLASMA_dgetrf</code> .
IPIV	<code>int*</code> (IN) The pivot indices from <code>PLASMA_dgetrf</code> (not equivalent to LAPACK).
B	<code>double*</code> (INOUT) On entry, the N-by-NRHS right hand side matrix B. On exit, if <code>return</code> value = 0, the N-by-NRHS solution matrix X.

28.3 Return Value:

```
= 0: successful exit
```

28.4 C Bindings

```
int PLASMA_dtrsml_Tile(PLASMA_desc *A, PLASMA_desc *L, int *IPIV, PLASMA_desc *B)
```

28.5 Fortran Bindings

```
void PLASMA_DTRSMPL_TILE(long long int *A, long long int *L, int **IPIVH, long long int *B, ↵
    int *INFO)
```

28.6 Online Browsing

Dive into [PLASMA_dtrsml_Tile](#)

29 PLASMA_sgelqf

29.1 Purpose

PLASMA_sgelqf - Computes the tile LQ factorization of a complex M-by-N matrix A: $A = L * Q$.

29.2 Arguments

M	<code>int</code> (IN) The number of rows of the matrix A. $M \geq 0$.
N	<code>int</code> (IN) The number of columns of the matrix A. $N \geq 0$.
A	<code>float*</code> (INOUT) On entry, the M-by-N matrix A. On exit, the elements on and below the diagonal of the array contain the m-by-min(↵ M,N) lower trapezoidal matrix L (L is lower triangular if $M \leq N$); the elements above ↵ the diagonal represent the unitary matrix Q as a product of elementary reflectors, ↵ stored by tiles.
LDA	<code>int</code> (IN) The leading dimension of the array A. $LDA \geq \max(1,M)$.
T	<code>float*</code> (OUT) On exit, auxiliary factorization data, required by PLASMA_sgelqs to solve the ↵ system of equations.

29.3 Return Value:

```
= 0: successful exit
< 0: if -i, the i-th argument had an illegal value
```

29.4 C Bindings

```
int PLASMA_sgelqf(int M, int N, float *A, int LDA, float *T)
```

29.5 Fortran Bindings

```
void PLASMA_SGELQF(int *M, int *N, float *A, int *LDA, float **T, int *INFO)
```

29.6 Online Browsing

Dive into [PLASMA_sgelqf](#)

30 PLASMA_sgelqf_Tile

30.1 Purpose

PLASMA_sgelqf_Tile - Computes the tile LQ factorization of a complex M-by-N matrix A: $A = L * Q$. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

30.2 Arguments

A	<p><code>float*</code> (INOUT)</p> <p>On entry, the M-by-N matrix A.</p> <p>On exit, the elements on and below the diagonal of the array contain the m-by-min(\leftarrow M,N) lower trapezoidal matrix L (L is lower triangular <code>if</code> M \leq N); the elements above \leftarrow the diagonal represent the unitary matrix Q as a product of elementary reflectors, \leftarrow stored by tiles.</p>
T	<p><code>float*</code> (OUT)</p> <p>On exit, auxiliary factorization data, required by PLASMA_sgelqs to solve the \leftarrow system of equations.</p>

30.3 Return Value:

`= 0`: successful exit

30.4 C Bindings

```
int PLASMA_sgelqf_Tile(PLASMA_desc *A, PLASMA_desc *T)
```

30.5 Fortran Bindings

```
void PLASMA_SGELQF_TILE(long long int *A, long long int *T, int *INFO)
```

30.6 Online Browsing

Dive into [PLASMA_sgelqf_Tile](#)

31 PLASMA_sgelqs

31.1 Purpose

PLASMA_sgelqs - Compute a minimum-norm solution $\min \|A*X - B\|$ using the LQ factorization $A = L*Q$ computed by PLASMA_sgelqf.

31.2 Arguments

M	<code>int</code> (IN) The number of rows of the matrix A. $M \geq 0$.
N	<code>int</code> (IN) The number of columns of the matrix A. $N \geq M \geq 0$.
NRHS	<code>int</code> (IN) The number of columns of B. $NRHS \geq 0$.
A	<code>float*</code> (IN) Details of the LQ factorization of the original matrix A as returned by <code>PLASMA_sgelqf</code> .
LDA	<code>int</code> (IN) The leading dimension of the array A. $LDA \geq M$.
T	<code>float*</code> (IN) Auxiliary factorization data, computed by PLASMA_sgelqf.
B	<code>float*</code> (INOUT) On entry, the M-by-NRHS right hand side matrix B. On exit, the N-by-NRHS solution matrix X.
LDB	<code>int</code> (IN) The leading dimension of the array B. $LDB \geq N$.

31.3 Return Value:

```
= 0: successful exit
< 0: if -i, the i-th argument had an illegal value
```

31.4 C Bindings

```
int PLASMA_sgelqs(int M, int N, int NRHS, float *A, int LDA, float *T, float *B, int LDB)
```

31.5 Fortran Bindings

```
void PLASMA_SGELQS(int *M, int *N, int *NRHS, float *A, int *LDA, float **T, float *B, int *LDB, int *INFO)
```

31.6 Online Browsing

Dive into [PLASMA_sgelqs](#)

32 PLASMA_sgelqs_Tile

32.1 Purpose

PLASMA_sgelqs_Tile - Compute a minimum-norm solution $\min \|A * X - B\|$ using the LQ factorization $A = L * Q$ computed by PLASMA_sgelqf_Tile. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

32.2 Arguments

A	<code>float*</code> (IN) Details of the LQ factorization of the original matrix A as returned by <code>PLASMA_sgelqf</code> .
T	<code>float*</code> (IN) Auxiliary factorization data, computed by <code>PLASMA_sgelqf</code> .
B	<code>float*</code> (INOUT) On entry, the M-by-NRHS right hand side matrix B. On exit, the N-by-NRHS solution matrix X.

32.3 Return Value:

`= 0`: successful exit

32.4 C Bindings

```
int PLASMA_sgelqs_Tile(PLASMA_desc *A, PLASMA_desc *B, PLASMA_desc *T)
```

32.5 Fortran Bindings

```
void PLASMA_SGELQS_TILE(long long int *A, long long int *B, long long int *T, int *INFO)
```

32.6 Online Browsing

Dive into [PLASMA_sgelqs_Tile](#)

33 PLASMA_sgels

33.1 Purpose

PLASMA_sgels - solves overdetermined or underdetermined linear systems involving an M-by-N matrix A using the QR or the LQ factorization of A. It is assumed that A has full rank. The following options are provided:

1. `trans = PlasmaNoTrans` and $M \geq N$: find the least squares solution of an overdetermined system, i.e., solve the least squares problem: minimize $\|B - A * X\|$.
2. `trans = PlasmaNoTrans` and $M < N$: find the minimum norm solution of an underdetermined system $A * X = B$.

Several right hand side vectors B and solution vectors X can be handled in a single call; they are stored as the columns of the M -by- $NRHS$ right hand side matrix B and the N -by- $NRHS$ solution matrix X .

33.2 Arguments

<code>trans</code>	<code>PLASMA_enum (IN)</code> Intended usage: <code>= PlasmaNoTrans</code> : the linear system involves A ; <code>= PlasmaTrans</code> : the linear system involves $A * T$. Currently only <code>PlasmaNoTrans</code> is supported.
<code>M</code>	<code>int (IN)</code> The number of rows of the matrix A . $M \geq 0$.
<code>N</code>	<code>int (IN)</code> The number of columns of the matrix A . $N \geq 0$.
<code>NRHS</code>	<code>int (IN)</code> The number of right hand sides, i.e., the number of columns of the matrices B and X . $NRHS \geq 0$.
<code>A</code>	<code>float* (INOUT)</code> On entry, the M -by- N matrix A . On exit, <code>if $M \geq N$</code> , A is overwritten by details of its QR factorization as returned by <code>PLASMA_sgeqrf</code> ; <code>if $M < N$</code> , A is overwritten by details of its LQ factorization as returned by <code>PLASMA_sgelqf</code> .
<code>LDA</code>	<code>int (IN)</code> The leading dimension of the array A . $LDA \geq \max(1, M)$.
<code>T</code>	<code>float* (OUT)</code> On exit, auxiliary factorization data.
<code>B</code>	<code>float* (INOUT)</code> On entry, the M -by- $NRHS$ matrix B of right hand side vectors, stored columnwise; On exit, <code>if return value = 0</code> , B is overwritten by the solution vectors, stored columnwise: <code>if $M \geq N$</code> , rows 1 to N of B contain the least squares solution vectors; the residual sum of squares for the solution in each column is given by the sum of squares of the modulus of elements $N+1$ to M in that column; <code>if $M < N$</code> , rows 1 to N of B contain the minimum norm solution vectors;
<code>LDB</code>	<code>int (IN)</code> The leading dimension of the array B . $LDB \geq \max(1, M, N)$.

33.3 Return Value:

`= 0`: successful exit
`< 0`: `if -i`, the i -th argument had an illegal value

33.4 C Bindings

```
int PLASMA_sgels(PLASMA_enum trans, int M, int N, int NRHS, float *A, int LDA, float *T, ←
                float *B, int LDB)
```

33.5 Fortran Bindings

```
void PLASMA_SGELS(PLASMA_enum *trans, int *M, int *N, int *NRHS, float *A, int *LDA, float ←
                 **T, float *B, int *LDB, int *INFO)
```

33.6 Online Browsing

Dive into [PLASMA_sgels](#)

34 PLASMA_sgels_Tile

34.1 Purpose

PLASMA_sgels_Tile - solves overdetermined or underdetermined linear systems involving an M-by-N matrix A using the QR or the LQ factorization of A. It is assumed that A has full rank. The following options are provided:

1. trans = PlasmaNoTrans and $M \geq N$: find the least squares solution of an overdetermined system, i.e., solve the least squares problem: minimize $\|B - A * X\|$.
2. trans = PlasmaNoTrans and $M < N$: find the minimum norm solution of an underdetermined system $A * X = B$.

Several right hand side vectors B and solution vectors X can be handled in a single call; they are stored as the columns of the M-by-NRHS right hand side matrix B and the N-by-NRHS solution matrix X. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

34.2 Arguments

trans	PLASMA_enum (IN) Intended usage: = PlasmaNoTrans: the linear system involves A; = PlasmaTrans: the linear system involves A**T. Currently only PlasmaNoTrans is supported.
A	float* (INOUT) On entry, the M-by-N matrix A. On exit, if $M \geq N$, A is overwritten by details of its QR factorization as returned by PLASMA_sgeqrf; if $M < N$, A is overwritten by details of its LQ factorization as returned by PLASMA_sgelqf.
T	float* (OUT) On exit, auxiliary factorization data.
B	float* (INOUT) On entry, the M-by-NRHS matrix B of right hand side vectors, stored columnwise; On exit, if return value = 0, B is overwritten by the solution vectors, stored columnwise:

```

if M >= N, rows 1 to N of B contain the least squares solution vectors; the
    residual
sum of squares for the solution in each column is given by the sum of squares of
    the
modulus of elements N+1 to M in that column;
if M < N, rows 1 to N of B contain the minimum norm solution vectors;

```

34.3 Return Value:

```

= 0: successful exit

```

34.4 C Bindings

```

int PLASMA_sgels_Tile(PLASMA_enum trans, PLASMA_desc *A, PLASMA_desc *B, PLASMA_desc *T)

```

34.5 Fortran Bindings

```

void PLASMA_SGELS_TILE(PLASMA_enum *trans, long long int *A, long long int *B, long long
    int *T, int *INFO)

```

34.6 Online Browsing

Dive into [PLASMA_sgels_Tile](#)

35 PLASMA_sgeqrf

35.1 Purpose

PLASMA_sgeqrf - Computes the tile QR factorization of a complex M-by-N matrix A: $A = Q * R$.

35.2 Arguments

M	<code>int</code> (IN) The number of rows of the matrix A. $M \geq 0$.
N	<code>int</code> (IN) The number of columns of the matrix A. $N \geq 0$.
A	<code>float*</code> (INOUT) On entry, the M-by-N matrix A. On exit, the elements on and above the diagonal of the array contain the min(M,N)- by-N upper trapezoidal matrix R (R is upper triangular if $M \geq N$); the elements below the diagonal represent the unitary matrix Q as a product of elementary reflectors stored by tiles.
LDA	<code>int</code> (IN) The leading dimension of the array A. $LDA \geq \max(1, M)$.

T `float*` (OUT)
 On exit, auxiliary factorization data, required by PLASMA_sgeqrs to solve the system of equations. ↵

35.3 Return Value:

= 0: successful exit
 < 0: if -i, the i-th argument had an illegal value

35.4 C Bindings

```
int PLASMA_sgeqrf(int M, int N, float *A, int LDA, float *T)
```

35.5 Fortran Bindings

```
void PLASMA_SGEQRF(int *M, int *N, float *A, int *LDA, float **T, int *INFO)
```

35.6 Online Browsing

Dive into [PLASMA_sgeqrf](#)

36 PLASMA_sgeqrf_Tile

36.1 Purpose

PLASMA_sgeqrf_Tile - Computes the tile QR factorization of a complex M-by-N matrix A: $A = Q * R$. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

36.2 Arguments

A `float*` (INOUT)
 On entry, the M-by-N matrix A.
 On exit, the elements on and above the diagonal of the array contain the min(M,N)-by-N upper trapezoidal matrix R (R is upper triangular if $M \geq N$); the elements below the diagonal represent the unitary matrix Q as a product of elementary reflectors stored by tiles. ↵

T `float*` (OUT)
 On exit, auxiliary factorization data, required by PLASMA_sgeqrs to solve the system of equations. ↵

36.3 Return Value:

```
= 0: successful exit
```

36.4 C Bindings

```
int PLASMA_sgeqrf_Tile(PLASMA_desc *A, PLASMA_desc *T)
```

36.5 Fortran Bindings

```
void PLASMA_SGEQRF_TILE(long long int *A, long long int *T, int *INFO)
```

36.6 Online Browsing

Dive into [PLASMA_sgeqrf_Tile](#)

37 PLASMA_sgeqrs

37.1 Purpose

PLASMA_sgeqrs - Compute a minimum-norm solution $\min \|A*X - B\|$ using the RQ factorization $A = R*Q$ computed by **PLASMA_sgeqrf**.

37.2 Arguments

M	<code>int</code> (IN) The number of rows of the matrix A. $M \geq 0$.
N	<code>int</code> (IN) The number of columns of the matrix A. $N \geq M \geq 0$.
NRHS	<code>int</code> (IN) The number of columns of B. $NRHS \geq 0$.
A	<code>float*</code> (INOUT) Details of the QR factorization of the original matrix A as returned by <code>↔</code> PLASMA_sgeqrf .
LDA	<code>int</code> (IN) The leading dimension of the array A. $LDA \geq M$.
T	<code>float*</code> (IN) Auxiliary factorization data, computed by PLASMA_sgeqrf .
B	<code>float*</code> (INOUT) On entry, the m-by-nrhs right hand side matrix B. On exit, the n-by-nrhs solution matrix X.
LDB	<code>int</code> (IN) The leading dimension of the array B. $LDB \geq \max(1, N)$.

37.3 Return Value:

```
= 0: successful exit
< 0: if -i, the i-th argument had an illegal value
```

37.4 C Bindings

```
int PLASMA_sgeqrs(int M, int N, int NRHS, float *A, int LDA, float *T, float *B, int LDB)
```

37.5 Fortran Bindings

```
void PLASMA_SGEQRS(int *M, int *N, int *NRHS, float *A, int *LDA, float **T, float *B, int ←
    *LDB, int *INFO)
```

37.6 Online Browsing

Dive into [PLASMA_sgeqrs](#)

38 PLASMA_sgeqrs_Tile

38.1 Purpose

PLASMA_sgeqrs_Tile - Compute a minimum-norm solution $\min \|A^*X - B\|$ using the RQ factorization $A = R^*Q$ computed by PLASMA_sgeqrf_Tile. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

38.2 Arguments

A	<code>float*</code> (INOUT) Details of the QR factorization of the original matrix A as returned by <code>←</code> PLASMA_sgeqrf.
T	<code>float*</code> (IN) Auxiliary factorization data, computed by PLASMA_sgeqrf.
B	<code>float*</code> (INOUT) On entry, the m-by-nrhs right hand side matrix B. On exit, the n-by-nrhs solution matrix X.

38.3 Return Value:

```
= 0: successful exit
```

38.4 C Bindings

```
int PLASMA_sgeqrs_Tile(PLASMA_desc *A, PLASMA_desc *B, PLASMA_desc *T)
```

38.5 Fortran Bindings

```
void PLASMA_SGEQRS_TILE(long long int *A, long long int *B, long long int *T, int *INFO)
```

38.6 Online Browsing

Dive into [PLASMA_sgeqrs_Tile](#)

39 PLASMA_sgesv

39.1 Purpose

PLASMA_sgesv - Computes the solution to a system of linear equations $A * X = B$, where A is an N -by- N matrix and X and B are N -by- $NRHS$ matrices. The tile LU decomposition with partial tile pivoting and row interchanges is used to factor A . The factored form of A is then used to solve the system of equations $A * X = B$.

39.2 Arguments

N	<code>int</code> (IN) The number of linear equations, i.e., the order of the matrix A . $N \geq 0$.
$NRHS$	<code>int</code> (IN) The number of right hand sides, i.e., the number of columns of the matrix B . $NRHS \geq 0$.
A	<code>float*</code> (INOUT) On entry, the N -by- N coefficient matrix A . On exit, the tile L and U factors from the factorization (not equivalent to LAPACK \leftrightarrow).
LDA	<code>int</code> (IN) The leading dimension of the array A . $LDA \geq \max(1, N)$.
L	<code>float*</code> (OUT) On exit, auxiliary factorization data, related to the tile L factor, necessary to solve the system of equations.
$IPIV$	<code>int*</code> (OUT) On exit, the pivot indices that define the permutations (not equivalent to LAPACK) \leftrightarrow .
B	<code>float*</code> (INOUT) On entry, the N -by- $NRHS$ matrix of right hand side matrix B . On exit, <code>if return value = 0</code> , the N -by- $NRHS$ solution matrix X .
LDB	<code>int</code> (IN) The leading dimension of the array B . $LDB \geq \max(1, N)$.

39.3 Return Value:

```
= 0: successful exit
< 0: if -i, the i-th argument had an illegal value
> 0: if i, U(i,i) is exactly zero. The factorization has been completed,
      but the factor U is exactly singular, so the solution could not be computed.
```

39.4 C Bindings

```
int PLASMA_sgesv(int N, int NRHS, float *A, int LDA, float *L, int *IPIV, float *B, int LDB ↵
)
```

39.5 Fortran Bindings

```
void PLASMA_SGESV(int *N, int *NRHS, float *A, int *LDA, float **LH, int **IPIVH, float *B, ↵
int *LDB, int *INFO)
```

39.6 Online Browsing

Dive into [PLASMA_sgesv](#)

40 PLASMA_sgesv_Tile

40.1 Purpose

PLASMA_sgesv_Tile - Computes the solution to a system of linear equations $A * X = B$, where A is an N -by- N matrix and X and B are N -by- $NRHS$ matrices. The tile LU decomposition with partial tile pivoting and row interchanges is used to factor A . The factored form of A is then used to solve the system of equations $A * X = B$. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

40.2 Arguments

A	<code>float*</code> (INOUT) On entry, the N -by- N coefficient matrix A . On exit, the tile L and U factors from the factorization (not equivalent to LAPACK ↵).
L	<code>float*</code> (OUT) On exit, auxiliary factorization data, related to the tile L factor, necessary to solve the system of equations.
IPIV	<code>int*</code> (OUT) On exit, the pivot indices that define the permutations (not equivalent to LAPACK) ↵ .
B	<code>float*</code> (INOUT) On entry, the N -by- $NRHS$ matrix of right hand side matrix B . On exit, <code>if return</code> value = 0, the N -by- $NRHS$ solution matrix X .

40.3 Return Value:

```
= 0: successful exit
> 0: if i, U(i,i) is exactly zero. The factorization has been completed,
    but the factor  $U$  is exactly singular, so the solution could not be computed.
```

40.4 C Bindings

```
int PLASMA_sgesv_Tile(PLASMA_desc *A, PLASMA_desc *L, int *IPIV, PLASMA_desc *B)
```

40.5 Fortran Bindings

```
void PLASMA_SGESV_TILE(long long int *A, long long int *L, int **IPIVH, long long int *B, ←  
int *INFO)
```

40.6 Online Browsing

Dive into [PLASMA_sgesv_Tile](#)

41 PLASMA_sgetrf

41.1 Purpose

PLASMA_sgetrf - Computes an LU factorization of a general M-by-N matrix A using the tile LU algorithm with partial tile pivoting with row interchanges.

41.2 Arguments

M	<code>int</code> (IN) The number of rows of the matrix A. $M \geq 0$.
N	<code>int</code> (IN) The number of columns of the matrix A. $N \geq 0$.
A	<code>float*</code> (INOUT) On entry, the M-by-N matrix to be factored. On exit, the tile factors L and U from the factorization.
LDA	<code>int</code> (IN) The leading dimension of the array A. $LDA \geq \max(1, M)$.
L	<code>float*</code> (OUT) On exit, auxiliary factorization data, related to the tile L factor, required by PLASMA_sgetrs to solve the system of equations.
IPIV	<code>int*</code> (OUT) The pivot indices that define the permutations (not equivalent to LAPACK).

41.3 Return Value:

```
= 0: successful exit  
< 0: if -i, the i-th argument had an illegal value  
> 0: if i, U(i,i) is exactly zero. The factorization has been completed,  
      but the factor U is exactly singular, and division by zero will occur  
      if it is used to solve a system of equations.
```

41.4 C Bindings

```
int PLASMA_sgetrf(int M, int N, float *A, int LDA, float *L, int *IPIV)
```

41.5 Fortran Bindings

```
void PLASMA_SGETRF(int *M, int *N, float *A, int *LDA, float **LH, int **IPIVH, int *INFO)
```

41.6 Online Browsing

Dive into [PLASMA_sgetrf](#)

42 PLASMA_sgetrf_Tile

42.1 Purpose

PLASMA_sgetrf_Tile - Computes an LU factorization of a general M-by-N matrix A using the tile LU algorithm with partial tile pivoting with row interchanges. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

42.2 Arguments

A	<code>float*</code> (INOUT) On entry, the M-by-N matrix to be factored. On exit, the tile factors L and U from the factorization.
L	<code>float*</code> (OUT) On exit, auxiliary factorization data, related to the tile L factor, required by PLASMA_sgetrs to solve the system of equations.
IPIV	<code>int*</code> (OUT) The pivot indices that define the permutations (not equivalent to LAPACK).

42.3 Return Value:

```
= 0: successful exit
> 0: if i, U(i,i) is exactly zero. The factorization has been completed,
      but the factor U is exactly singular, and division by zero will occur
      if it is used to solve a system of equations.
```

42.4 C Bindings

```
int PLASMA_sgetrf_Tile(PLASMA_desc *A, PLASMA_desc *L, int *IPIV)
```

42.5 Fortran Bindings

```
void PLASMA_SGETRF_TILE(long long int *A, long long int *L, int **IPIVH, int *INFO)
```

42.6 Online Browsing

Dive into [PLASMA_sgetrf_Tile](#)

43 PLASMA_sgetrs

43.1 Purpose

PLASMA_sgetrs - Solves a system of linear equations $A * X = B$, with a general N-by-N matrix A using the tile LU factorization computed by PLASMA_sgetrf.

43.2 Arguments

trans	PLASMA_enum (IN) Intended to specify the the form of the system of equations: = PlasmaNoTrans: $A * X = B$ (No transpose) = PlasmaTrans: $A^{**T} * X = B$ (Transpose) = PlasmaTrans: $A^{**T} * X = B$ (Conjugate transpose) Currently only PlasmaNoTrans is supported.
N	<code>int</code> (IN) The order of the matrix A. $N \geq 0$.
NRHS	<code>int</code> (IN) The number of right hand sides, i.e., the number of columns of the matrix B. $NRHS \geq 0$.
A	<code>float*</code> (IN) The tile factors L and U from the factorization, computed by PLASMA_sgetrf.
LDA	<code>int</code> (IN) The leading dimension of the array A. $LDA \geq \max(1, N)$.
L	<code>float*</code> (IN) Auxiliary factorization data, related to the tile L factor, computed by \leftrightarrow PLASMA_sgetrf.
IPIV	<code>int*</code> (IN) The pivot indices from PLASMA_sgetrf (not equivalent to LAPACK).
B	<code>float*</code> (INOUT) On entry, the N-by-NRHS matrix of right hand side matrix B. On exit, the solution matrix X.
LDB	<code>int</code> (IN) The leading dimension of the array B. $LDB \geq \max(1, N)$.

43.3 Return Value:

```
= 0: successful exit
< 0: if -i, the i-th argument had an illegal value
```

43.4 C Bindings

```
int PLASMA_sgetrs(PLASMA_enum uplo, int N, int NRHS, float *A, int LDA, float *L, int *IPIV ↵
, float *B, int LDB)
```

43.5 Fortran Bindings

```
void PLASMA_SGETRS(PLASMA_enum *uplo, int *N, int *NRHS, float *A, int *LDA, float **LH, ↵
int **IPIVH, float *B, int *LDB, int *INFO)
```

43.6 Online Browsing

Dive into [PLASMA_sgetrs](#)

44 PLASMA_sgetrs_Tile

44.1 Purpose

PLASMA_sgetrs_Tile - Solves a system of linear equations $A * X = B$, with a general N-by-N matrix A using the tile LU factorization computed by PLASMA_sgetrf. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

44.2 Arguments

A	<code>float*</code> (IN) The tile factors L and U from the factorization, computed by PLASMA_sgetrf.
L	<code>float*</code> (IN) Auxiliary factorization data, related to the tile L factor, computed by ↵ PLASMA_sgetrf.
IPIV	<code>int*</code> (IN) The pivot indices from PLASMA_sgetrf (not equivalent to LAPACK).
B	<code>float*</code> (INOUT) On entry, the N-by-NRHS matrix of right hand side matrix B. On exit, the solution matrix X.

44.3 Return Value:

= 0: successful exit

44.4 C Bindings

```
int PLASMA_sgetrs_Tile(PLASMA_desc *A, PLASMA_desc *L, int *IPIV, PLASMA_desc *B)
```

44.5 Fortran Bindings

```
void PLASMA_SGETRS_TILE(long long int *A, long long int *L, int **IPIVH, long long int *B,
    int *INFO)
```

44.6 Online Browsing

Dive into [PLASMA_sgetrs_Tile](#)

45 PLASMA_sposv

45.1 Purpose

PLASMA_sposv - Computes the solution to a system of linear equations $A * X = B$, where A is an N-by-N symmetric positive definite (or Hermitian positive definite in the complex case) matrix and X and B are N-by-NRHS matrices. The Cholesky decomposition is used to factor A as

$A = U^{*}T * U$, if `uplo = PlasmaUpper`, or
 $A = L * L^{*}T$, if `uplo = PlasmaLower`,

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 $A * X = B$.

45.2 Arguments

uplo	PLASMA_enum (IN) Specifies whether the matrix A is upper triangular or lower triangular: = PlasmaUpper: Upper triangle of A is stored; = PlasmaLower: Lower triangle of A is stored.
N	int (IN) The number of linear equations, i.e., the order of the matrix A. $N \geq 0$.
NRHS	int (IN) The number of right hand sides, i.e., the number of columns of the matrix B. NRHS ≥ 0 .
A	float* (INOUT) On entry, the symmetric positive definite (or Hermitian) matrix A. If <code>uplo = PlasmaUpper</code> , the leading N-by-N upper triangular part of A contains the upper triangular part of the matrix A, and the strictly lower triangular part of A is not referenced. If <code>UPLO = 'L'</code> , the leading N-by-N lower triangular part of A contains the lower triangular part of the matrix A, and the strictly upper triangular part of A is not referenced. On exit, if <code>return value = 0</code> , the factor U or L from the Cholesky factorization $A = U^{*}T*U$ or $A = L*L^{*}T$.
LDA	int (IN) The leading dimension of the array A. $LDA \geq \max(1, N)$.
B	float* (INOUT)

On entry, the N-by-NRHS right hand side matrix B.
 On exit, if return value = 0, the N-by-NRHS solution matrix X.

LDB `int` (IN)
 The leading dimension of the array B. LDB $\geq \max(1, N)$.

45.3 Return Value:

```
= 0: successful exit
< 0: if -i, the i-th argument had an illegal value
> 0: if i, the leading minor of order i of A is not positive definite, so the
      factorization could not be completed, and the solution has not been computed ↵
      .
```

45.4 C Bindings

```
int PLASMA_sposv(PLASMA_enum uplo, int N, int NRHS, float *A, int LDA, float *B, int LDB)
```

45.5 Fortran Bindings

```
void PLASMA_SPOSV(PLASMA_enum *uplo, int *N, int *NRHS, float *A, int *LDA, float *B, int * ↵
  LDB, int *INFO)
```

45.6 Online Browsing

Dive into [PLASMA_sposv](#)

46 PLASMA_sposv_Tile

46.1 Purpose

PLASMA_sposv_Tile - Computes the solution to a system of linear equations $A * X = B$, where A is an N-by-N symmetric positive definite (or Hermitian positive definite in the complex case) matrix and X and B are N-by-NRHS matrices. The Cholesky decomposition is used to factor A as

$A = U^{*T} * U$, if uplo = PlasmaUpper, or
 $A = L * L^{*T}$, if uplo = PlasmaLower,

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 $A * X = B$. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

46.2 Arguments

uplo PLASMA_enum (IN)
 Specifies whether the matrix A is upper triangular or lower triangular:
 = PlasmaUpper: Upper triangle of A is stored;
 = PlasmaLower: Lower triangle of A is stored.

A `float*` (INOUT)

On entry, the symmetric positive definite (or Hermitian) matrix A.
 If uplo = PlasmaUpper, the leading N-by-N upper triangular part of A contains the upper triangular part of the matrix A, and the strictly lower triangular part of A is not referenced.
 If UPLO = 'L', the leading N-by-N lower triangular part of A contains the lower triangular part of the matrix A, and the strictly upper triangular part of A is not referenced.
 On exit, if return value = 0, the factor U or L from the Cholesky factorization $A = U^*T^*U$ or $A = L^*L^*T$.
 B float* (INOUT)
 On entry, the N-by-NRHS right hand side matrix B.
 On exit, if return value = 0, the N-by-NRHS solution matrix X.

46.3 Return Value:

= 0: successful exit
 > 0: if i, the leading minor of order i of A is not positive definite, so the factorization could not be completed, and the solution has not been computed
 .

46.4 C Bindings

```
int PLASMA_sposv_Tile(PLASMA_enum uplo, PLASMA_desc *A, PLASMA_desc *B)
```

46.5 Fortran Bindings

```
void PLASMA_SPOSV_TILE(PLASMA_enum *uplo, long long int *A, long long int *B, int *INFO)
```

46.6 Online Browsing

Dive into [PLASMA_sposv_Tile](#)

47 PLASMA_spotrf

47.1 Purpose

PLASMA_spotrf - Computes the Cholesky factorization of a symmetric positive definite (or Hermitian positive definite in the complex case) matrix A. The factorization has the form

$A = U^*T^*U$, if uplo = PlasmaUpper, or
 $A = L^*L^*T$, if uplo = PlasmaLower,

where U is an upper triangular matrix and L is a lower triangular matrix.

47.2 Arguments

```

uplo    PLASMA_enum (IN)
        = PlasmaUpper: Upper triangle of A is stored;
        = PlasmaLower: Lower triangle of A is stored.

N       int (IN)
        The order of the matrix A. N >= 0.

A       float* (INOUT)
        On entry, the symmetric positive definite (or Hermitian) matrix A.
        If uplo = PlasmaUpper, the leading N-by-N upper triangular part of A
        contains the upper triangular part of the matrix A, and the strictly lower ↵
        triangular
        part of A is not referenced.
        If UPLO = 'L', the leading N-by-N lower triangular part of A contains the lower
        triangular part of the matrix A, and the strictly upper triangular part of A is ↵
        not
        referenced.
        On exit, if return value = 0, the factor U or L from the Cholesky factorization
        A = U*T*U or A = L*L*T.

LDA     int (IN)
        The leading dimension of the array A. LDA >= max(1,N).

```

47.3 Return Value:

```

= 0: successful exit
< 0: if -i, the i-th argument had an illegal value
> 0: if i, the leading minor of order i of A is not positive definite, so the
      factorization could not be completed, and the solution has not been computed ↵
      .

```

47.4 C Bindings

```
int PLASMA_spotrf(PLASMA_enum uplo, int N, float *A, int LDA)
```

47.5 Fortran Bindings

```
void PLASMA_SPOTRF(PLASMA_enum *uplo, int *N, float *A, int *LDA, int *INFO)
```

47.6 Online Browsing

Dive into [PLASMA_spotrf](#)

48 PLASMA_spotrf_Tile

48.1 Purpose

PLASMA_spotrf_Tile - Computes the Cholesky factorization of a symmetric positive definite (or Hermitian positive definite in the complex case) matrix A. The factorization has the form

$A = U^{*T} * U$, if `uplo = PlasmaUpper`, or
 $A = L * L^{*T}$, if `uplo = PlasmaLower`,

where U is an upper triangular matrix and L is a lower triangular matrix. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

48.2 Arguments

```
uplo      PLASMA_enum (IN)
          = PlasmaUpper: Upper triangle of A is stored;
          = PlasmaLower: Lower triangle of A is stored.

A          float* (INOUT)
          On entry, the symmetric positive definite (or Hermitian) matrix A.
          If uplo = PlasmaUpper, the leading N-by-N upper triangular part of A
          contains the upper triangular part of the matrix A, and the strictly lower ↵
          triangular
          part of A is not referenced.
          If UPLO = 'L', the leading N-by-N lower triangular part of A contains the lower
          triangular part of the matrix A, and the strictly upper triangular part of A is ↵
          not
          referenced.
          On exit, if return value = 0, the factor U or L from the Cholesky factorization
          A = U^{*T}*U or A = L*L^{*T}.
```

48.3 Return Value:

```
= 0: successful exit
> 0: if i, the leading minor of order i of A is not positive definite, so the
    factorization could not be completed, and the solution has not been computed ↵
    .
```

48.4 C Bindings

```
int PLASMA_spotrf_Tile(PLASMA_enum uplo, PLASMA_desc *A)
```

48.5 Fortran Bindings

```
void PLASMA_SPOTRF_TILE(PLASMA_enum *uplo, long long int *A, int *INFO)
```

48.6 Online Browsing

Dive into [PLASMA_spotrf_Tile](#)

49 PLASMA_spotrs

49.1 Purpose

PLASMA_spotrs - Solves a system of linear equations $A * X = B$ with a symmetric positive definite (or Hermitian positive definite in the complex case) matrix A using the Cholesky factorization $A = UT^{*}U$ or $A = L^{*}LT$ computed by `PLASMA_spotrf`.

49.2 Arguments

uplo	PLASMA_enum (IN) = PlasmaUpper: Upper triangle of A is stored; = PlasmaLower: Lower triangle of A is stored.
N	int (IN) The order of the matrix A. $N \geq 0$.
NRHS	int (IN) The number of right hand sides, i.e., the number of columns of the matrix B. $NRHS \geq 0$.
A	float* (IN) The triangular factor U or L from the Cholesky factorization $A = U^*T^*U$ or $A = L^*L$ computed by PLASMA_spotrf.
LDA	int (IN) The leading dimension of the array A. $LDA \geq \max(1, N)$.
B	float* (INOUT) On entry, the N-by-NRHS right hand side matrix B. On exit, if return value = 0, the N-by-NRHS solution matrix X.
LDB	int (IN) The leading dimension of the array B. $LDB \geq \max(1, N)$.

49.3 Return Value:

= 0: successful exit
< 0: if -i, the i-th argument had an illegal value

49.4 C Bindings

```
int PLASMA_spotrs(PLASMA_enum uplo, int N, int NRHS, float *A, int LDA, float *B, int LDB)
```

49.5 Fortran Bindings

```
void PLASMA_SPOTRS(PLASMA_enum *uplo, int *N, int *NRHS, float *A, int *LDA, float *B, int* ←  
LDB, int * INFO)
```

49.6 Online Browsing

Dive into [PLASMA_spotrs](#)

50 PLASMA_spotrs_Tile

50.1 Purpose

PLASMA_spotrs_Tile - Solves a system of linear equations $A * X = B$ with a symmetric positive definite (or Hermitian positive definite in the complex case) matrix A using the Cholesky factorization $A = UT^*U$ or $A = L^*LT$ computed by PLASMA_spotrf. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

50.2 Arguments

uplo	PLASMA_enum (IN) = PlasmaUpper: Upper triangle of A is stored; = PlasmaLower: Lower triangle of A is stored.
A	float* (IN) The triangular factor U or L from the Cholesky factorization $A = U^*T^*U$ or $A = L^*L \leftrightarrow U^*T^*$, computed by PLASMA_spotrf.
B	float* (INOUT) On entry, the N-by-NRHS right hand side matrix B. On exit, if return value = 0, the N-by-NRHS solution matrix X.

50.3 Return Value:

= 0: successful exit

50.4 C Bindings

```
int PLASMA_spotrs_Tile(PLASMA_enum uplo, PLASMA_desc *A, PLASMA_desc *B)
```

50.5 Fortran Bindings

```
void PLASMA_SPOTRS_TILE(PLASMA_enum *uplo, long long int *A, long long int *B, int *INFO)
```

50.6 Online Browsing

Dive into [PLASMA_spotrs_Tile](#)

51 PLASMA_strsm

51.1 Purpose

PLASMA_strsm - Computes triangular solve $A^*X = B$ or $X^*A = B$

51.2 Arguments

side	PLASMA_enum (IN) Specifies whether A appears on the left or on the right of X: = PlasmaLeft: $A^*X = B$ = PlasmaRight: $X^*A = B$
uplo	PLASMA_enum (IN) Specifies whether the matrix A is upper triangular or lower triangular: = PlasmaUpper: Upper triangle of A is stored; = PlasmaLower: Lower triangle of A is stored.
transA	PLASMA_enum (IN)

	Specifies whether the matrix A is transposed, not transposed or conjugate transposed: = PlasmaNoTrans: A is transposed; = PlasmaTrans: A is not transposed; = PlasmaTrans: A is conjugate transposed.
diag	PLASMA_enum (IN) Specifies whether or not A is unit triangular: = PlasmaNonUnit: A is non unit; = PlasmaUnit: A is unit.
N	int (IN) The order of the matrix A. N >= 0.
NRHS	int (IN) The number of right hand sides, i.e., the number of columns of the matrix B. NRHS >= 0.
A	float* (IN) The triangular matrix A. If uplo = PlasmaUpper, the leading N-by-N upper triangular part of the array A contains the upper triangular matrix, and the strictly lower triangular part of A is not referenced. If uplo = PlasmaLower, the leading N-by-N lower triangular part of the array A contains the lower triangular matrix, and the strictly upper triangular part of A is not referenced. If diag = PlasmaUnit, the diagonal elements of A are also not referenced and are assumed to be 1.
LDA	int (IN) The leading dimension of the array A. LDA >= max(1,N).
B	float* (INOUT) On entry, the N-by-NRHS right hand side matrix B. On exit, if return value = 0, the N-by-NRHS solution matrix X.
LDB	float* (IN) The leading dimension of the array B. LDB >= max(1,N).

51.3 Return Value:

```

= 0: successful exit
< 0: if -i, the i-th argument had an illegal value

```

51.4 C Bindings

```

int PLASMA_strsm(PLASMA_enum side, PLASMA_enum uplo, PLASMA_enum transA, PLASMA_enum diag,
int N, int NRHS, float *A, int LDA, float *B, int LDB)

```

51.5 Fortran Bindings

```

void PLASMA_STRSM(PLASMA_enum *side, PLASMA_enum *uplo, PLASMA_enum *transA, PLASMA_enum *
diag, int *N, int *NRHS, float *A, int *LDA, float *B, int *LDB, int *INFO)

```

51.6 Online Browsing

Dive into [PLASMA_strsm](#)

52 PLASMA_strsm_Tile

52.1 Purpose

PLASMA_strsm_Tile - Computes triangular solve $A*X = B$ or $X*A = B$. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

52.2 Arguments

side	PLASMA_enum (IN) Specifies whether A appears on the left or on the right of X: = PlasmaLeft: $A*X = B$ = PlasmaRight: $X*A = B$
uplo	PLASMA_enum (IN) Specifies whether the matrix A is upper triangular or lower triangular: = PlasmaUpper: Upper triangle of A is stored; = PlasmaLower: Lower triangle of A is stored.
transA	PLASMA_enum (IN) Specifies whether the matrix A is transposed, not transposed or conjugate transposed: \leftrightarrow = PlasmaNoTrans: A is transposed; = PlasmaTrans: A is not transposed; = PlasmaTrans: A is conjugate transposed.
diag	PLASMA_enum (IN) Specifies whether or not A is unit triangular: = PlasmaNonUnit: A is non unit; = PlasmaUnit: A is unit.
A	float* (IN) The triangular matrix A. If uplo = PlasmaUpper, the leading N-by-N upper triangular part of the array A contains the upper triangular matrix, and the strictly lower triangular part of A is not referenced. If uplo = PlasmaLower, the leading N-by-N lower triangular part of the array A contains the lower triangular matrix, and the strictly upper triangular part of A is not referenced. If diag = PlasmaUnit, the diagonal elements of A are also not referenced and are assumed to be 1. \leftrightarrow
B	float* (INOUT) On entry, the N-by-NRHS right hand side matrix B. On exit, if return value = 0, the N-by-NRHS solution matrix X.

52.3 Return Value:

= 0: successful exit

52.4 C Bindings

```
int PLASMA_strsm_Tile(PLASMA_enum side, PLASMA_enum uplo, PLASMA_enum transA, PLASMA_enum diag, PLASMA_desc *A, PLASMA_desc *B)  $\leftrightarrow$ 
```

52.5 Fortran Bindings

```
void PLASMA_STRSM_TILE(PLASMA_enum *side, PLASMA_enum *uplo, PLASMA_enum *transA, ↵
    PLASMA_enum *diag, long long int *A, long long int *B, int *INFO)
```

52.6 Online Browsing

Dive into [PLASMA_strsm_Tile](#)

53 PLASMA_strsmpl

53.1 Purpose

PLASMA_strsmpl - Performs the forward substitution step of solving a system of linear equations after the tile LU factorization of the matrix.

53.2 Arguments

N	<code>int</code> (IN) The order of the matrix A. $N \geq 0$.
NRHS	<code>int</code> (IN) The number of right hand sides, i.e., the number of columns of the matrix B. NRHS ↵ ≥ 0 .
A	<code>float*</code> (IN) The tile factor L from the factorization, computed by <code>PLASMA_sgetrf</code> .
LDA	<code>int</code> (IN) The leading dimension of the array A. $LDA \geq \max(1, N)$.
L	<code>float*</code> (IN) Auxiliary factorization data, related to the tile L factor, computed by ↵ <code>PLASMA_sgetrf</code> .
IPIV	<code>int*</code> (IN) The pivot indices from <code>PLASMA_sgetrf</code> (not equivalent to LAPACK).
B	<code>float*</code> (INOUT) On entry, the N-by-NRHS right hand side matrix B. On exit, <code>if return</code> value = 0, the N-by-NRHS solution matrix X.
LDB	<code>float*</code> (IN) The leading dimension of the array B. $LDB \geq \max(1, N)$.

53.3 Return Value:

```
= 0: successful exit
< 0: if -i, the i-th argument had an illegal value
```

53.4 C Bindings

```
int PLASMA_strsmpl(int N, int NRHS, float *A, int LDA, float *L, int *IPIV, float *B, int ←
LDB)
```

53.5 Fortran Bindings

```
void PLASMA_STRSMPL(int *N, int *NRHS, float *A, int *LDA, float **LH, int **IPIVH, float * ←
B, int *LDB, int *INFO)
```

53.6 Online Browsing

Dive into [PLASMA_strsmpl](#)

54 PLASMA_strsmpl_Tile

54.1 Purpose

PLASMA_strsmpl_Tile - Performs the forward substitution step of solving a system of linear equations after the tile LU factorization of the matrix. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

54.2 Arguments

A	<code>float*</code> (IN) The tile factor L from the factorization, computed by PLASMA_sgetrf.
L	<code>float*</code> (IN) Auxiliary factorization data, related to the tile L factor, computed by ← PLASMA_sgetrf.
IPIV	<code>int*</code> (IN) The pivot indices from PLASMA_sgetrf (not equivalent to LAPACK).
B	<code>float*</code> (INOUT) On entry, the N-by-NRHS right hand side matrix B. On exit, <code>if return</code> value = 0, the N-by-NRHS solution matrix X.

54.3 Return Value:

= 0: successful exit

54.4 C Bindings

```
int PLASMA_strsmpl_Tile(PLASMA_desc *A, PLASMA_desc *L, int *IPIV, PLASMA_desc *B)
```

54.5 Fortran Bindings

```
void PLASMA_STRSMPL_TILE(long long int *A, long long int *L, int **IPIVH, long long int *B, ←
int *INFO)
```

54.6 Online Browsing

Dive into [PLASMA_strsmpl_Tile](#)

55 PLASMA_zcgesv

55.1 Purpose

PLASMA_zcgesv - Computes the solution to a system of linear equations $A * X = B$, where A is an N -by- N matrix and X and B are N -by- $NRHS$ matrices. The tile LU decomposition with partial tile pivoting and row interchanges is used to factor A . The factored form of A is then used to solve the system of equations $A * X = B$.

IMPORTANT NOTICE: in its current state, this routine only intends to be a proof-of-concept. There are still some costly serial parts and one may NOT expect to achieve high performance.

PLASMA_zcgesv first attempts to factorize the matrix in COMPLEX and use this factorization within an iterative refinement procedure to produce a solution with COMPLEX*16 normwise backward error quality (see below). If the approach fails the method switches to a COMPLEX*16 factorization and solve.

The iterative refinement is not going to be a winning strategy if the ratio COMPLEX performance over COMPLEX*16 performance is too small. A reasonable strategy should take the number of right-hand sides and the size of the matrix into account. This might be done with a call to ILAENV in the future. Up to now, we always try iterative refinement.

The iterative refinement process is stopped if $ITER > ITERMAX$ or for all the RHS we have: $RNRM < SQRT(N)*XNRM*ANRM*EPS*$ where

- $ITER$ is the number of the current iteration in the iterative refinement process
- $RNRM$ is the infinity-norm of the residual
- $XNRM$ is the infinity-norm of the solution
- $ANRM$ is the infinity-operator-norm of the matrix A
- EPS is the machine epsilon returned by `DLAMCH(Epsilon)`.

The values $ITERMAX$ and $BWDMAX$ are fixed to 30 and 1.0D+00 respectively.

55.2 Arguments

N	<code>int</code> (IN) The number of linear equations, i.e., the order of the matrix A . $N \geq 0$.
$NRHS$	<code>int</code> (IN) The number of right hand sides, i.e., the number of columns of the matrix B . $NRHS \geq 0$.
A	<code>PLASMA_Complex64_t*</code> (INOUT) On entry, the N -by- N coefficient matrix A . On exit, the tile L and U factors from the factorization (not equivalent to LAPACK \leftrightarrow).
LDA	<code>int</code> (IN) The leading dimension of the array A . $LDA \geq \max(1, N)$.
L	<code>PLASMA_Complex64_t*</code> (OUT) On exit, auxiliary factorization data, related to the tile L factor, necessary to solve the system of equations.

```

IPIV      int* (OUT)
          On exit, the pivot indices that define the permutations (not equivalent to LAPACK) ←
          .

B         PLASMA_Complex64_t* (IN)
          The N-by-NRHS matrix of right hand side matrix B.

LDB       int (IN)
          The leading dimension of the array B. LDB >= max(1,N).

X         PLASMA_Complex64_t* (OUT)
          If return value = 0, the N-by-NRHS solution matrix X.

LDX       int (IN)
          The leading dimension of the array B. LDX >= max(1,N).

ITER      int* (OUT) is the number of the current iteration in the iterative refinement ←
          process

```

55.3 Return Value:

```

= 0: successful exit
< 0: if -i, the i-th argument had an illegal value
> 0: if i, U(i,i) is exactly zero. The factorization has been completed,
      but the factor U is exactly singular, so the solution could not be computed.

```

55.4 C Bindings

```

int PLASMA_zcgesv(int N, int NRHS, PLASMA_Complex64_t *A, int LDA, PLASMA_Complex64_t *L, ←
int *IPIV, PLASMA_Complex64_t *B, int LDB, PLASMA_Complex64_t *X, int LDX, int *ITER)

```

55.5 Fortran Bindings

```

void PLASMA_ZCGESV(int *N, int *NRHS, PLASMA_Complex64_t *A, int *LDA, PLASMA_Complex64_t ←
**LH, int **IPIVH, PLASMA_Complex64_t *B, int *LDB, PLASMA_Complex64_t *X, int *LDX, int ←
*ITER, int *INFO)

```

55.6 Online Browsing

Dive into [PLASMA_zcgesv](#)

56 PLASMA_zcgesv_Tile

56.1 Purpose

PLASMA_zcgesv - Computes the solution to a system of linear equations $A * X = B$, where A is an N -by- N matrix and X and B are N -by- $NRHS$ matrices. The tile LU decomposition with partial tile pivoting and row interchanges is used to factor A . The factored form of A is then used to solve the system of equations $A * X = B$. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

IMPORTANT NOTICE: in its current state, this routine only intends to be a proof-of-concept. There are still some costly serial parts and one may NOT expect to achieve high performance.

PLASMA_zcgesv first attempts to factorize the matrix in COMPLEX and use this factorization within an iterative refinement procedure to produce a solution with COMPLEX*16 normwise backward error quality (see below). If the approach fails the method switches to a COMPLEX*16 factorization and solve.

The iterative refinement is not going to be a winning strategy if the ratio COMPLEX performance over COMPLEX*16 performance is too small. A reasonable strategy should take the number of right-hand sides and the size of the matrix into account. This might be done with a call to ILAENV in the future. Up to now, we always try iterative refinement.

The iterative refinement process is stopped if $ITER > ITERMAX$ or for all the RHS we have: $RNRM < \sqrt{N} * XNRM * ANRM * EPS$ where

- $ITER$ is the number of the current iteration in the iterative refinement process
- $RNRM$ is the infinity-norm of the residual
- $XNRM$ is the infinity-norm of the solution
- $ANRM$ is the infinity-operator-norm of the matrix A
- EPS is the machine epsilon returned by $DLAMCH(Epsilon)$.

The values $ITERMAX$ and $BWDMAX$ are fixed to 30 and 1.0D+00 respectively.

56.2 Arguments

N	<code>int</code> (IN) The number of linear equations, i.e., the order of the matrix A. $N \geq 0$.
NRHS	<code>int</code> (IN) The number of right hand sides, i.e., the number of columns of the matrix B. $NRHS \geq 0$.
A	PLASMA_Complex64_t* (INOUT) On entry, the N-by-N coefficient matrix A. On exit, the tile L and U factors from the factorization (not equivalent to LAPACK ←).
LDA	<code>int</code> (IN) The leading dimension of the array A. $LDA \geq \max(1, N)$.
L	PLASMA_Complex64_t* (OUT) On exit, auxiliary factorization data, related to the tile L factor, necessary to solve the system of equations.
IPIV	<code>int*</code> (OUT) On exit, the pivot indices that define the permutations (not equivalent to LAPACK) ←.
B	PLASMA_Complex64_t* (IN) The N-by-NRHS matrix of right hand side matrix B.
LDB	<code>int</code> (IN) The leading dimension of the array B. $LDB \geq \max(1, N)$.
X	PLASMA_Complex64_t* (OUT) If <code>return</code> value = 0, the N-by-NRHS solution matrix X.
LDX	<code>int</code> (IN) The leading dimension of the array B. $LDX \geq \max(1, N)$.
ITER	<code>int*</code> (OUT) is the number of the current iteration in the iterative refinement process ←

56.3 Return Value:

```
= 0: successful exit
< 0: if -i, the i-th argument had an illegal value
> 0: if i, U(i,i) is exactly zero. The factorization has been completed,
      but the factor U is exactly singular, so the solution could not be computed.
```

56.4 C Bindings

```
int PLASMA_zcgesv_Tile(PLASMA_desc *A, PLASMA_desc *L, int *IPIV, PLASMA_desc *B, ←
    PLASMA_desc *X, int *ITER)
```

56.5 Fortran Bindings

```
void PLASMA_ZCGESV_TILE(long long int *A, long long int *L, int **IPIVH, long long int *B, ←
    long long *X, int *ITER, int *INFO)
```

56.6 Online Browsing

Dive into [PLASMA_zcgesv_Tile](#)

57 PLASMA_zgelqf

57.1 Purpose

PLASMA_zgelqf - Computes the tile LQ factorization of a complex M-by-N matrix A: $A = L * Q$.

57.2 Arguments

M	<code>int</code> (IN) The number of rows of the matrix A. $M \geq 0$.
N	<code>int</code> (IN) The number of columns of the matrix A. $N \geq 0$.
A	<code>PLASMA_Complex64_t*</code> (INOUT) On entry, the M-by-N matrix A. On exit, the elements on and below the diagonal of the array contain the m-by-min(← M,N) lower trapezoidal matrix L (L is lower triangular if $M \leq N$); the elements above ← the diagonal represent the unitary matrix Q as a product of elementary reflectors, ← stored by tiles.
LDA	<code>int</code> (IN) The leading dimension of the array A. $LDA \geq \max(1, M)$.
T	<code>PLASMA_Complex64_t*</code> (OUT) On exit, auxiliary factorization data, required by <code>PLASMA_zgelqs</code> to solve the ← system of equations.

57.3 Return Value:

```
= 0: successful exit
< 0: if -i, the i-th argument had an illegal value
```

57.4 C Bindings

```
int PLASMA_zgelqf(int M, int N, PLASMA_Complex64_t *A, int LDA, PLASMA_Complex64_t *T)
```

57.5 Fortran Bindings

```
void PLASMA_ZGELQF(int *M, int *N, PLASMA_Complex64_t *A, int *LDA, PLASMA_Complex64_t **T, ↵
int *INFO)
```

57.6 Online Browsing

Dive into [PLASMA_zgelqf](#)

58 PLASMA_zgelqf_Tile

58.1 Purpose

PLASMA_zgelqf_Tile - Computes the tile LQ factorization of a complex M-by-N matrix A: $A = L * Q$. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

58.2 Arguments

A	PLASMA_Complex64_t* (INOUT) On entry, the M-by-N matrix A. On exit, the elements on and below the diagonal of the array contain the m-by-min(↵ M,N) lower trapezoidal matrix L (L is lower triangular if $M \leq N$); the elements above ↵ the diagonal represent the unitary matrix Q as a product of elementary reflectors, ↵ stored by tiles.
T	PLASMA_Complex64_t* (OUT) On exit, auxiliary factorization data, required by PLASMA_zgelqs to solve the ↵ system of equations.

58.3 Return Value:

```
= 0: successful exit
```

58.4 C Bindings

```
int PLASMA_zgelqf_Tile(PLASMA_desc *A, PLASMA_desc *T)
```

58.5 Fortran Bindings

```
void PLASMA_ZGELQF_TILE(long long int *A, long long int *T, int *INFO)
```

58.6 Online Browsing

Dive into [PLASMA_zgelqf_Tile](#)

59 PLASMA_zgelqs

59.1 Purpose

PLASMA_zgelqs - Compute a minimum-norm solution $\min \|A*X - B\|$ using the LQ factorization $A = L*Q$ computed by **PLASMA_zgelqf**.

59.2 Arguments

M	<code>int</code> (IN) The number of rows of the matrix A. $M \geq 0$.
N	<code>int</code> (IN) The number of columns of the matrix A. $N \geq M \geq 0$.
NRHS	<code>int</code> (IN) The number of columns of B. $NRHS \geq 0$.
A	<code>PLASMA_Complex64_t*</code> (IN) Details of the LQ factorization of the original matrix A as returned by <code>PLASMA_zgelqf</code> .
LDA	<code>int</code> (IN) The leading dimension of the array A. $LDA \geq M$.
T	<code>PLASMA_Complex64_t*</code> (IN) Auxiliary factorization data, computed by <code>PLASMA_zgelqf</code> .
B	<code>PLASMA_Complex64_t*</code> (INOUT) On entry, the M-by-NRHS right hand side matrix B. On exit, the N-by-NRHS solution matrix X.
LDB	<code>int</code> (IN) The leading dimension of the array B. $LDB \geq N$.

59.3 Return Value:

```
= 0: successful exit
< 0: if -i, the i-th argument had an illegal value
```

59.4 C Bindings

```
int PLASMA_zgelqs(int M, int N, int NRHS, PLASMA_Complex64_t *A, int LDA, ↵
    PLASMA_Complex64_t *T, PLASMA_Complex64_t *B, int LDB)
```

59.5 Fortran Bindings

```
void PLASMA_ZGELQS(int *M, int *N, int *NRHS, PLASMA_Complex64_t *A, int *LDA, ↵
    PLASMA_Complex64_t **T, PLASMA_Complex64_t *B, int *LDB, int *INFO)
```

59.6 Online Browsing

Dive into [PLASMA_zgelqs](#)

60 PLASMA_zgelqs_Tile

60.1 Purpose

PLASMA_zgelqs_Tile - Compute a minimum-norm solution $\min \|A^*X - B\|$ using the LQ factorization $A = L^*Q$ computed by `PLASMA_zgelqf_Tile`. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

60.2 Arguments

A	PLASMA_Complex64_t* (IN) Details of the LQ factorization of the original matrix A as returned by ↵ PLASMA_zgelqf.
T	PLASMA_Complex64_t* (IN) Auxiliary factorization data, computed by PLASMA_zgelqf.
B	PLASMA_Complex64_t* (INOUT) On entry, the M-by-NRHS right hand side matrix B. On exit, the N-by-NRHS solution matrix X.

60.3 Return Value:

= 0: successful exit

60.4 C Bindings

```
int PLASMA_zgelqs_Tile(PLASMA_desc *A, PLASMA_desc *B, PLASMA_desc *T)
```

60.5 Fortran Bindings

```
void PLASMA_ZGELQS_TILE(long long int *A, long long int *B, long long int *T, int *INFO)
```

60.6 Online Browsing

Dive into [PLASMA_zgelqs_Tile](#)

61 PLASMA_zgels

61.1 Purpose

PLASMA_zgels - solves overdetermined or underdetermined linear systems involving an M-by-N matrix A using the QR or the LQ factorization of A. It is assumed that A has full rank. The following options are provided:

1. `trans = PlasmaNoTrans` and $M \geq N$: find the least squares solution of an overdetermined system, i.e., solve the least squares problem: minimize $\|B - A * X\|$.
2. `trans = PlasmaNoTrans` and $M < N$: find the minimum norm solution of an underdetermined system $A * X = B$.

Several right hand side vectors B and solution vectors X can be handled in a single call; they are stored as the columns of the M-by-NRHS right hand side matrix B and the N-by-NRHS solution matrix X.

61.2 Arguments

<code>trans</code>	PLASMA_enum (IN) Intended usage: = <code>PlasmaNoTrans</code> : the linear system involves A; = <code>PlasmaConjTrans</code> : the linear system involves A^*H . Currently only <code>PlasmaNoTrans</code> is supported.
<code>M</code>	int (IN) The number of rows of the matrix A. $M \geq 0$.
<code>N</code>	int (IN) The number of columns of the matrix A. $N \geq 0$.
<code>NRHS</code>	int (IN) The number of right hand sides, i.e., the number of columns of the matrices B and \leftrightarrow X. $NRHS \geq 0$.
<code>A</code>	PLASMA_Complex64_t* (INOUT) On entry, the M-by-N matrix A. On exit, if $M \geq N$, A is overwritten by details of its QR factorization as returned by <code>PLASMA_zgeqrf</code> ; if $M < N$, A is overwritten by details of its LQ factorization as returned by <code>PLASMA_zgelqf</code> .
<code>LDA</code>	int (IN) The leading dimension of the array A. $LDA \geq \max(1, M)$.
<code>T</code>	PLASMA_Complex64_t* (OUT) On exit, auxiliary factorization data.
<code>B</code>	PLASMA_Complex64_t* (INOUT) On entry, the M-by-NRHS matrix B of right hand side vectors, stored columnwise; On exit, if <code>return</code> value = 0, B is overwritten by the solution vectors, stored columnwise: if $M \geq N$, rows 1 to N of B contain the least squares solution vectors; the \leftrightarrow residual

```

        sum of squares for the solution in each column is given by the sum of squares of
        the
        modulus of elements N+1 to M in that column;
        if M < N, rows 1 to N of B contain the minimum norm solution vectors;

LDB      int (IN)
        The leading dimension of the array B. LDB >= MAX(1,M,N).

```

61.3 Return Value:

```

    = 0: successful exit
    < 0: if -i, the i-th argument had an illegal value

```

61.4 C Bindings

```

int PLASMA_zgels(PLASMA_enum trans, int M, int N, int NRHS, PLASMA_Complex64_t *A, int LDA,
    PLASMA_Complex64_t *T, PLASMA_Complex64_t *B, int LDB)

```

61.5 Fortran Bindings

```

void PLASMA_ZGELS(PLASMA_enum *trans, int *M, int *N, int *NRHS, PLASMA_Complex64_t *A, int
    *LDA, PLASMA_Complex64_t **T, PLASMA_Complex64_t *B, int *LDB, int *INFO)

```

61.6 Online Browsing

Dive into [PLASMA_zgels](#)

62 PLASMA_zgels_Tile

62.1 Purpose

PLASMA_zgels_Tile - solves overdetermined or underdetermined linear systems involving an M-by-N matrix A using the QR or the LQ factorization of A. It is assumed that A has full rank. The following options are provided:

1. trans = PlasmaNoTrans and M >= N: find the least squares solution of an overdetermined system, i.e., solve the least squares problem: minimize $\|B - A * X\|$.
2. trans = PlasmaNoTrans and M < N: find the minimum norm solution of an underdetermined system $A * X = B$.

Several right hand side vectors B and solution vectors X can be handled in a single call; they are stored as the columns of the M-by-NRHS right hand side matrix B and the N-by-NRHS solution matrix X. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

62.2 Arguments

trans	PLASMA_enum (IN) Intended usage: = PlasmaNoTrans: the linear system involves A; = PlasmaConjTrans: the linear system involves A**H. Currently only PlasmaNoTrans is supported.
A	PLASMA_Complex64_t* (INOUT) On entry, the M-by-N matrix A. On exit, if $M \geq N$, A is overwritten by details of its QR factorization as returned by PLASMA_zgeqrf; if $M < N$, A is overwritten by details of its LQ factorization as returned by PLASMA_zgelqf.
T	PLASMA_Complex64_t* (OUT) On exit, auxiliary factorization data.
B	PLASMA_Complex64_t* (INOUT) On entry, the M-by-NRHS matrix B of right hand side vectors, stored columnwise; On exit, if return value = 0, B is overwritten by the solution vectors, stored columnwise: if $M \geq N$, rows 1 to N of B contain the least squares solution vectors; the residual sum of squares for the solution in each column is given by the sum of squares of the modulus of elements N+1 to M in that column; if $M < N$, rows 1 to N of B contain the minimum norm solution vectors;

62.3 Return Value:

= 0: successful exit

62.4 C Bindings

```
int PLASMA_zgels_Tile(PLASMA_enum trans, PLASMA_desc *A, PLASMA_desc *B, PLASMA_desc *T)
```

62.5 Fortran Bindings

```
void PLASMA_ZGELS_TILE(PLASMA_enum *trans, long long int *A, long long int *B, long long  
int *T, int *INFO)
```

62.6 Online Browsing

Dive into [PLASMA_zgels_Tile](#)

63 PLASMA_zgeqrf

63.1 Purpose

PLASMA_zgeqrf - Computes the tile QR factorization of a complex M-by-N matrix A: $A = Q * R$.

63.2 Arguments

M	<code>int</code> (IN) The number of rows of the matrix A. $M \geq 0$.
N	<code>int</code> (IN) The number of columns of the matrix A. $N \geq 0$.
A	<code>PLASMA_Complex64_t*</code> (INOUT) On entry, the M-by-N matrix A. On exit, the elements on and above the diagonal of the array contain the min(M,N)-by-N upper trapezoidal matrix R (R is upper triangular <code>if</code> $M \geq N$); the elements below the diagonal represent the unitary matrix Q as a product of elementary reflectors stored by tiles.
LDA	<code>int</code> (IN) The leading dimension of the array A. $LDA \geq \max(1, M)$.
T	<code>PLASMA_Complex64_t*</code> (OUT) On exit, auxiliary factorization data, required by <code>PLASMA_zgeqrs</code> to solve the system of equations.

63.3 Return Value:

= 0: successful exit
< 0: `if -i`, the i-th argument had an illegal value

63.4 C Bindings

```
int PLASMA_zgeqrf(int M, int N, PLASMA_Complex64_t *A, int LDA, PLASMA_Complex64_t *T)
```

63.5 Fortran Bindings

```
void PLASMA_ZGEQRF(int *M, int *N, PLASMA_Complex64_t *A, int *LDA, PLASMA_Complex64_t **T, int *INFO)
```

63.6 Online Browsing

Dive into [PLASMA_zgeqrf](#)

64 PLASMA_zgeqrf_Tile

64.1 Purpose

PLASMA_zgeqrf_Tile - Computes the tile QR factorization of a complex M-by-N matrix A: $A = Q * R$. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

64.2 Arguments

A	PLASMA_Complex64_t* (INOUT) On entry, the M-by-N matrix A. On exit, the elements on and above the diagonal of the array contain the min(M,N)-by-N upper trapezoidal matrix R (R is upper triangular if $M \geq N$); the elements below the diagonal represent the unitary matrix Q as a product of elementary reflectors stored by tiles.
T	PLASMA_Complex64_t* (OUT) On exit, auxiliary factorization data, required by PLASMA_zgeqrs to solve the system of equations.

64.3 Return Value:

= 0: successful exit

64.4 C Bindings

```
int PLASMA_zgeqrf_Tile(PLASMA_desc *A, PLASMA_desc *T)
```

64.5 Fortran Bindings

```
void PLASMA_ZGEQRF_TILE(long long int *A, long long int *T, int *INFO)
```

64.6 Online Browsing

Dive into [PLASMA_zgeqrf_Tile](#)

65 PLASMA_zgeqrs

65.1 Purpose

PLASMA_zgeqrs - Compute a minimum-norm solution $\min \|A^*X - B\|$ using the RQ factorization $A = R^*Q$ computed by PLASMA_zgeqrf.

65.2 Arguments

M	int (IN) The number of rows of the matrix A. $M \geq 0$.
N	int (IN) The number of columns of the matrix A. $N \geq M \geq 0$.
NRHS	int (IN)

	The number of columns of B. NRHS ≥ 0 .
A	PLASMA_Complex64_t* (INOUT) Details of the QR factorization of the original matrix A as returned by <code>PLASMA_zgeqrf</code> .
LDA	<code>int</code> (IN) The leading dimension of the array A. LDA $\geq M$.
T	PLASMA_Complex64_t* (IN) Auxiliary factorization data, computed by <code>PLASMA_zgeqrf</code> .
B	PLASMA_Complex64_t* (INOUT) On entry, the m-by-nrhs right hand side matrix B. On exit, the n-by-nrhs solution matrix X.
LDB	<code>int</code> (IN) The leading dimension of the array B. LDB $\geq \max(1, N)$.

65.3 Return Value:

```
= 0: successful exit
< 0: if -i, the i-th argument had an illegal value
```

65.4 C Bindings

```
int PLASMA_zgeqrs(int M, int N, int NRHS, PLASMA_Complex64_t *A, int LDA,
    PLASMA_Complex64_t *T, PLASMA_Complex64_t *B, int LDB)
```

65.5 Fortran Bindings

```
void PLASMA_ZGEQRS(int *M, int *N, int *NRHS, PLASMA_Complex64_t *A, int *LDA,
    PLASMA_Complex64_t **T, PLASMA_Complex64_t *B, int *LDB, int *INFO)
```

65.6 Online Browsing

Dive into [PLASMA_zgeqrs](#)

66 PLASMA_zgeqrs_Tile

66.1 Purpose

PLASMA_zgeqrs_Tile - Compute a minimum-norm solution $\min \|A^*X - B\|$ using the RQ factorization $A = R^*Q$ computed by `PLASMA_zgeqrf_Tile`. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

66.2 Arguments

A	PLASMA_Complex64_t* (INOUT) Details of the QR factorization of the original matrix A as returned by <code>PLASMA_zgeqrf</code> .
T	PLASMA_Complex64_t* (IN) Auxiliary factorization data, computed by <code>PLASMA_zgeqrf</code> .
B	PLASMA_Complex64_t* (INOUT) On entry, the m-by-nrhs right hand side matrix B. On exit, the n-by-nrhs solution matrix X.

66.3 Return Value:

= 0: successful exit

66.4 C Bindings

```
int PLASMA_zgeqrs_Tile(PLASMA_desc *A, PLASMA_desc *B, PLASMA_desc *T)
```

66.5 Fortran Bindings

```
void PLASMA_ZGEQRS_TILE(long long int *A, long long int *B, long long int *T, int *INFO)
```

66.6 Online Browsing

Dive into [PLASMA_zgeqrs_Tile](#)

67 PLASMA_zgesv

67.1 Purpose

PLASMA_zgesv - Computes the solution to a system of linear equations $A * X = B$, where A is an N-by-N matrix and X and B are N-by-NRHS matrices. The tile LU decomposition with partial tile pivoting and row interchanges is used to factor A. The factored form of A is then used to solve the system of equations $A * X = B$.

67.2 Arguments

N	<code>int</code> (IN) The number of linear equations, i.e., the order of the matrix A. $N \geq 0$.
NRHS	<code>int</code> (IN) The number of right hand sides, i.e., the number of columns of the matrix B. $NRHS \geq 0$.
A	PLASMA_Complex64_t* (INOUT) On entry, the N-by-N coefficient matrix A. On exit, the tile L and U factors from the factorization (not equivalent to LAPACK <code>←</code>).

LDA	<code>int</code> (IN) The leading dimension of the array A. $LDA \geq \max(1, N)$.
L	<code>PLASMA_Complex64_t*</code> (OUT) On exit, auxiliary factorization data, related to the tile L factor, necessary to solve the system of equations.
IPIV	<code>int*</code> (OUT) On exit, the pivot indices that define the permutations (not equivalent to LAPACK) \leftrightarrow .
B	<code>PLASMA_Complex64_t*</code> (INOUT) On entry, the N-by-NRHS matrix of right hand side matrix B. On exit, <code>if return</code> value = 0, the N-by-NRHS solution matrix X.
LDB	<code>int</code> (IN) The leading dimension of the array B. $LDB \geq \max(1, N)$.

67.3 Return Value:

```

= 0: successful exit
< 0: if -i, the i-th argument had an illegal value
> 0: if i, U(i,i) is exactly zero. The factorization has been completed,
    but the factor U is exactly singular, so the solution could not be computed.

```

67.4 C Bindings

```

int PLASMA_zgesv(int N, int NRHS, PLASMA_Complex64_t *A, int LDA, PLASMA_Complex64_t *L,  $\leftrightarrow$ 
int *IPIV, PLASMA_Complex64_t *B, int LDB)

```

67.5 Fortran Bindings

```

void PLASMA_ZGESV(int *N, int *NRHS, PLASMA_Complex64_t *A, int *LDA, PLASMA_Complex64_t **  $\leftrightarrow$ 
LH, int **IPIVH, PLASMA_Complex64_t *B, int *LDB, int *INFO)

```

67.6 Online Browsing

Dive into [PLASMA_zgesv](#)

68 PLASMA_zgesv_Tile

68.1 Purpose

PLASMA_zgesv_Tile - Computes the solution to a system of linear equations $A * X = B$, where A is an N-by-N matrix and X and B are N-by-NRHS matrices. The tile LU decomposition with partial tile pivoting and row interchanges is used to factor A. The factored form of A is then used to solve the system of equations $A * X = B$. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

68.2 Arguments

A	PLASMA_Complex64_t* (INOUT) On entry, the N-by-N coefficient matrix A. On exit, the tile L and U factors from the factorization (not equivalent to LAPACK ↔).
L	PLASMA_Complex64_t* (OUT) On exit, auxiliary factorization data, related to the tile L factor, necessary to solve the system of equations.
IPIV	int* (OUT) On exit, the pivot indices that define the permutations (not equivalent to LAPACK) ↔.
B	PLASMA_Complex64_t* (INOUT) On entry, the N-by-NRHS matrix of right hand side matrix B. On exit, if return value = 0, the N-by-NRHS solution matrix X.

68.3 Return Value:

```

= 0: successful exit
> 0: if i, U(i,i) is exactly zero. The factorization has been completed,
      but the factor U is exactly singular, so the solution could not be computed.

```

68.4 C Bindings

```
int PLASMA_zgesv_Tile(PLASMA_desc *A, PLASMA_desc *L, int *IPIV, PLASMA_desc *B)
```

68.5 Fortran Bindings

```
void PLASMA_ZGESV_TILE(long long int *A, long long int *L, int **IPIVH, long long int *B, ←
int *INFO)
```

68.6 Online Browsing

Dive into [PLASMA_zgesv_Tile](#)

69 PLASMA_zgetrf

69.1 Purpose

PLASMA_zgetrf - Computes an LU factorization of a general M-by-N matrix A using the tile LU algorithm with partial tile pivoting with row interchanges.

69.2 Arguments

M	<code>int</code> (IN) The number of rows of the matrix A. $M \geq 0$.
N	<code>int</code> (IN) The number of columns of the matrix A. $N \geq 0$.
A	<code>PLASMA_Complex64_t*</code> (INOUT) On entry, the M-by-N matrix to be factored. On exit, the tile factors L and U from the factorization.
LDA	<code>int</code> (IN) The leading dimension of the array A. $LDA \geq \max(1, M)$.
L	<code>PLASMA_Complex64_t*</code> (OUT) On exit, auxiliary factorization data, related to the tile L factor, required by <code>PLASMA_zgetrs</code> to solve the system of equations.
IPIV	<code>int*</code> (OUT) The pivot indices that define the permutations (not equivalent to LAPACK).

69.3 Return Value:

```

= 0: successful exit
< 0: if -i, the i-th argument had an illegal value
> 0: if i, U(i,i) is exactly zero. The factorization has been completed,
      but the factor U is exactly singular, and division by zero will occur
      if it is used to solve a system of equations.

```

69.4 C Bindings

```

int PLASMA_zgetrf(int M, int N, PLASMA_Complex64_t *A, int LDA, PLASMA_Complex64_t *L, int ←
    *IPIV)

```

69.5 Fortran Bindings

```

void PLASMA_ZGETRF(int *M, int *N, PLASMA_Complex64_t *A, int *LDA, PLASMA_Complex64_t **LH ←
    , int **IPIVH, int *INFO)

```

69.6 Online Browsing

Dive into [PLASMA_zgetrf](#)

70 PLASMA_zgetrf_Tile

70.1 Purpose

PLASMA_zgetrf_Tile - Computes an LU factorization of a general M-by-N matrix A using the tile LU algorithm with partial tile pivoting with row interchanges. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

70.2 Arguments

A	PLASMA_Complex64_t* (INOUT) On entry, the M-by-N matrix to be factored. On exit, the tile factors L and U from the factorization.
L	PLASMA_Complex64_t* (OUT) On exit, auxiliary factorization data, related to the tile L factor, required by PLASMA_zgetrs to solve the system of equations.
IPIV	int* (OUT) The pivot indices that define the permutations (not equivalent to LAPACK).

70.3 Return Value:

```

= 0: successful exit
> 0: if i, U(i,i) is exactly zero. The factorization has been completed,
      but the factor U is exactly singular, and division by zero will occur
      if it is used to solve a system of equations.

```

70.4 C Bindings

```
int PLASMA_zgetrf_Tile(PLASMA_desc *A, PLASMA_desc *L, int *IPIV)
```

70.5 Fortran Bindings

```
void PLASMA_ZGETRF_TILE(long long int *A, long long int *L, int **IPIVH, int *INFO)
```

70.6 Online Browsing

Dive into [PLASMA_zgetrf_Tile](#)

71 PLASMA_zgetrs

71.1 Purpose

PLASMA_zgetrs - Solves a system of linear equations $A * X = B$, with a general N-by-N matrix A using the tile LU factorization computed by PLASMA_zgetrf.

71.2 Arguments

trans	PLASMA_enum (IN) Intended to specify the the form of the system of equations: = PlasmaNoTrans: $A * X = B$ (No transpose) = PlasmaTrans: $A^{**T} * X = B$ (Transpose) = PlasmaConjTrans: $A^{**H} * X = B$ (Conjugate transpose) Currently only PlasmaNoTrans is supported.
N	int (IN)

	The order of the matrix A. $N \geq 0$.
NRHS	<code>int</code> (IN) The number of right hand sides, i.e., the number of columns of the matrix B. $NRHS \geq 0$.
A	<code>PLASMA_Complex64_t*</code> (IN) The tile factors L and U from the factorization, computed by <code>PLASMA_zgetrf</code> .
LDA	<code>int</code> (IN) The leading dimension of the array A. $LDA \geq \max(1, N)$.
L	<code>PLASMA_Complex64_t*</code> (IN) Auxiliary factorization data, related to the tile L factor, computed by <code>PLASMA_zgetrf</code> .
IPIV	<code>int*</code> (IN) The pivot indices from <code>PLASMA_zgetrf</code> (not equivalent to LAPACK).
B	<code>PLASMA_Complex64_t*</code> (INOUT) On entry, the N-by-NRHS matrix of right hand side matrix B. On exit, the solution matrix X.
LDB	<code>int</code> (IN) The leading dimension of the array B. $LDB \geq \max(1, N)$.

71.3 Return Value:

```

= 0: successful exit
< 0: if -i, the i-th argument had an illegal value

```

71.4 C Bindings

```

int PLASMA_zgetrs(PLASMA_enum uplo, int N, int NRHS, PLASMA_Complex64_t *A, int LDA,
PLASMA_Complex64_t *L, int *IPIV, PLASMA_Complex64_t *B, int LDB)

```

71.5 Fortran Bindings

```

void PLASMA_ZGETRS(PLASMA_enum *uplo, int *N, int *NRHS, PLASMA_Complex64_t *A, int *LDA,
PLASMA_Complex64_t **LH, int **IPIVH, PLASMA_Complex64_t *B, int *LDB, int *INFO)

```

71.6 Online Browsing

Dive into [PLASMA_zgetrs](#)

72 PLASMA_zgetrs_Tile

72.1 Purpose

PLASMA_zgetrs_Tile - Solves a system of linear equations $A * X = B$, with a general N-by-N matrix A using the tile LU factorization computed by `PLASMA_zgetrf`. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

72.2 Arguments

A	PLASMA_Complex64_t* (IN) The tile factors L and U from the factorization, computed by PLASMA_zgetrf.
L	PLASMA_Complex64_t* (IN) Auxiliary factorization data, related to the tile L factor, computed by \leftrightarrow PLASMA_zgetrf.
IPIV	int* (IN) The pivot indices from PLASMA_zgetrf (not equivalent to LAPACK).
B	PLASMA_Complex64_t* (INOUT) On entry, the N-by-NRHS matrix of right hand side matrix B. On exit, the solution matrix X.

72.3 Return Value:

= 0: successful exit

72.4 C Bindings

```
int PLASMA_zgetrs_Tile(PLASMA_desc *A, PLASMA_desc *L, int *IPIV, PLASMA_desc *B)
```

72.5 Fortran Bindings

```
void PLASMA_ZGETRS_TILE(long long int *A, long long int *L, int **IPIVH, long long int *B,  $\leftrightarrow$   
int *INFO)
```

72.6 Online Browsing

Dive into [PLASMA_zgetrs_Tile](#)

73 PLASMA_zposv

73.1 Purpose

PLASMA_zposv - Computes the solution to a system of linear equations $A * X = B$, where A is an N-by-N symmetric positive definite (or Hermitian positive definite in the complex case) matrix and X and B are N-by-NRHS matrices. The Cholesky decomposition is used to factor A as

$A = U^{*H} * U$, if uplo = PlasmaUpper, or
 $A = L * L^{*H}$, if uplo = PlasmaLower,

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 $A * X = B$.

73.2 Arguments

uplo	PLASMA_enum (IN) Specifies whether the matrix A is upper triangular or lower triangular: = PlasmaUpper: Upper triangle of A is stored; = PlasmaLower: Lower triangle of A is stored.
N	int (IN) The number of linear equations, i.e., the order of the matrix A. $N \geq 0$.
NRHS	int (IN) The number of right hand sides, i.e., the number of columns of the matrix B. NRHS ≥ 0 .
A	PLASMA_Complex64_t* (INOUT) On entry, the symmetric positive definite (or Hermitian) matrix A. If uplo = PlasmaUpper, the leading N-by-N upper triangular part of A contains the upper triangular part of the matrix A, and the strictly lower triangular part of A is not referenced. If UPLO = 'L', the leading N-by-N lower triangular part of A contains the lower triangular part of the matrix A, and the strictly upper triangular part of A is not referenced. On exit, if return value = 0, the factor U or L from the Cholesky factorization $A = U^*H^*U$ or $A = L^*L^*H$.
LDA	int (IN) The leading dimension of the array A. $LDA \geq \max(1, N)$.
B	PLASMA_Complex64_t* (INOUT) On entry, the N-by-NRHS right hand side matrix B. On exit, if return value = 0, the N-by-NRHS solution matrix X.
LDB	int (IN) The leading dimension of the array B. $LDB \geq \max(1, N)$.

73.3 Return Value:

```

= 0: successful exit
< 0: if -i, the i-th argument had an illegal value
> 0: if i, the leading minor of order i of A is not positive definite, so the
      factorization could not be completed, and the solution has not been computed

```

73.4 C Bindings

```

int PLASMA_zposv(PLASMA_enum uplo, int N, int NRHS, PLASMA_Complex64_t *A, int LDA,
  PLASMA_Complex64_t *B, int LDB)

```

73.5 Fortran Bindings

```

void PLASMA_ZPOSV(PLASMA_enum *uplo, int *N, int *NRHS, PLASMA_Complex64_t *A, int *LDA,
  PLASMA_Complex64_t *B, int *LDB, int *INFO)

```

73.6 Online Browsing

Dive into [PLASMA_zposv](#)

74 PLASMA_zposv_Tile

74.1 Purpose

PLASMA_zposv_Tile - Computes the solution to a system of linear equations $A * X = B$, where A is an N -by- N symmetric positive definite (or Hermitian positive definite in the complex case) matrix and X and B are N -by- $NRHS$ matrices. The Cholesky decomposition is used to factor A as

$$A = U^{*H} * U, \text{ if } \text{uplo} = \text{PlasmaUpper}, \text{ or}$$

$$A = L * L^{*H}, \text{ if } \text{uplo} = \text{PlasmaLower},$$

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 $A * X = B$. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

74.2 Arguments

uplo	PLASMA_enum (IN) Specifies whether the matrix A is upper triangular or lower triangular: = PlasmaUpper: Upper triangle of A is stored; = PlasmaLower: Lower triangle of A is stored.
A	PLASMA_Complex64_t* (INOUT) On entry, the symmetric positive definite (or Hermitian) matrix A . If uplo = PlasmaUpper, the leading N -by- N upper triangular part of A contains the upper triangular part of the matrix A , and the strictly lower triangular part of A is not referenced. ↔ If UPLO = 'L', the leading N -by- N lower triangular part of A contains the lower triangular part of the matrix A , and the strictly upper triangular part of A is not referenced. On exit, if return value = 0, the factor U or L from the Cholesky factorization $A = U^{*H} * U$ or $A = L * L^{*H}$.
B	PLASMA_Complex64_t* (INOUT) On entry, the N -by- $NRHS$ right hand side matrix B . On exit, if return value = 0, the N -by- $NRHS$ solution matrix X .

74.3 Return Value:

```
= 0: successful exit
> 0: if i, the leading minor of order i of A is not positive definite, so the
    factorization could not be completed, and the solution has not been computed ↔
.
```

74.4 C Bindings

```
int PLASMA_zposv_Tile(PLASMA_enum uplo, PLASMA_desc *A, PLASMA_desc *B)
```

74.5 Fortran Bindings

```
void PLASMA_ZPOSV_TILE(PLASMA_enum *uplo, long long int *A, long long int *B, int *INFO)
```

74.6 Online Browsing

Dive into [PLASMA_zposv_Tile](#)

75 PLASMA_zpotrf

75.1 Purpose

PLASMA_zpotrf - Computes the Cholesky factorization of a symmetric positive definite (or Hermitian positive definite in the complex case) matrix A. The factorization has the form

$A = U^{*}H * U$, if uplo = PlasmaUpper, or
 $A = L * L^{*}H$, if uplo = PlasmaLower,

where U is an upper triangular matrix and L is a lower triangular matrix.

75.2 Arguments

uplo	PLASMA_enum (IN) = PlasmaUpper: Upper triangle of A is stored; = PlasmaLower: Lower triangle of A is stored.
N	int (IN) The order of the matrix A. $N \geq 0$.
A	PLASMA_Complex64_t* (INOUT) On entry, the symmetric positive definite (or Hermitian) matrix A. If uplo = PlasmaUpper, the leading N-by-N upper triangular part of A contains the upper triangular part of the matrix A, and the strictly lower triangular part of A is not referenced. If UPLO = 'L', the leading N-by-N lower triangular part of A contains the lower triangular part of the matrix A, and the strictly upper triangular part of A is not referenced. On exit, if return value = 0, the factor U or L from the Cholesky factorization $A = U^{*}H * U$ or $A = L * L^{*}H$.
LDA	int (IN) The leading dimension of the array A. $LDA \geq \max(1, N)$.

75.3 Return Value:

= 0: successful exit
 < 0: if -i, the i-th argument had an illegal value
 > 0: if i, the leading minor of order i of A is not positive definite, so the factorization could not be completed, and the solution has not been computed ↵

75.4 C Bindings

```
int PLASMA_zpotrf(PLASMA_enum uplo, int N, PLASMA_Complex64_t *A, int LDA)
```

75.5 Fortran Bindings

```
void PLASMA_ZPOTRF(PLASMA_enum *uplo, int *N, PLASMA_Complex64_t *A, int *LDA, int *INFO)
```

75.6 Online Browsing

Dive into [PLASMA_zpotrf](#)

76 PLASMA_zpotrf_Tile

76.1 Purpose

PLASMA_zpotrf_Tile - Computes the Cholesky factorization of a symmetric positive definite (or Hermitian positive definite in the complex case) matrix A. The factorization has the form

$A = U^{*H} * U$, if uplo = PlasmaUpper, or
 $A = L * L^{*H}$, if uplo = PlasmaLower,

where U is an upper triangular matrix and L is a lower triangular matrix. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

76.2 Arguments

uplo	PLASMA_enum (IN) = PlasmaUpper: Upper triangle of A is stored; = PlasmaLower: Lower triangle of A is stored.
A	PLASMA_Complex64_t* (INOUT) On entry, the symmetric positive definite (or Hermitian) matrix A. If uplo = PlasmaUpper, the leading N-by-N upper triangular part of A contains the upper triangular part of the matrix A, and the strictly lower ↔ triangular part of A is not referenced. If UPLO = 'L', the leading N-by-N lower triangular part of A contains the lower triangular part of the matrix A, and the strictly upper triangular part of A is ↔ not referenced. On exit, if return value = 0, the factor U or L from the Cholesky factorization $A = U^{*H} * U$ or $A = L * L^{*H}$.

76.3 Return Value:

= 0: successful exit
 > 0: if i, the leading minor of order i of A is not positive definite, so the factorization could not be completed, and the solution has not been computed ↔
 .

76.4 C Bindings

```
int PLASMA_zpotrf_Tile(PLASMA_enum uplo, PLASMA_desc *A)
```

76.5 Fortran Bindings

```
void PLASMA_ZPOTRF_TILE(PLASMA_enum *uplo, long long int *A, int *INFO)
```

76.6 Online Browsing

Dive into [PLASMA_zpotrf_Tile](#)

77 PLASMA_zpotrs

77.1 Purpose

PLASMA_zpotrs - Solves a system of linear equations $A * X = B$ with a symmetric positive definite (or Hermitian positive definite in the complex case) matrix A using the Cholesky factorization $A = UH*U$ or $A = L*LH$ computed by **PLASMA_zpotrf**.

77.2 Arguments

uplo	PLASMA_enum (IN) = PlasmaUpper: Upper triangle of A is stored; = PlasmaLower: Lower triangle of A is stored.
N	int (IN) The order of the matrix A. $N \geq 0$.
NRHS	int (IN) The number of right hand sides, i.e., the number of columns of the matrix B. NRHS $\leftarrow \geq 0$.
A	PLASMA_Complex64_t* (IN) The triangular factor U or L from the Cholesky factorization $A = U*H*U$ or $A = L*L$ $\leftarrow **H$, computed by PLASMA_zpotrf .
LDA	int (IN) The leading dimension of the array A. $LDA \geq \max(1, N)$.
B	PLASMA_Complex64_t* (INOUT) On entry, the N-by-NRHS right hand side matrix B. On exit, if return value = 0, the N-by-NRHS solution matrix X.
LDB	int (IN) The leading dimension of the array B. $LDB \geq \max(1, N)$.

77.3 Return Value:

```
= 0: successful exit
< 0: if -i, the i-th argument had an illegal value
```

77.4 C Bindings

```
int PLASMA_zpotrs(PLASMA_enum uplo, int N, int NRHS, PLASMA_Complex64_t *A, int LDA, ↵
    PLASMA_Complex64_t *B, int LDB)
```

77.5 Fortran Bindings

```
void PLASMA_ZPOTRS(PLASMA_enum *uplo, int *N, int *NRHS, PLASMA_Complex64_t *A, int *LDA, ↵
    PLASMA_Complex64_t *B, int* LDB, int * INFO)
```

77.6 Online Browsing

Dive into [PLASMA_zpotrs](#)

78 PLASMA_zpotrs_Tile

78.1 Purpose

PLASMA_zpotrs_Tile - Solves a system of linear equations $A * X = B$ with a symmetric positive definite (or Hermitian positive definite in the complex case) matrix A using the Cholesky factorization $A = UH*U$ or $A = L*LH$ computed by **PLASMA_zpotrf**. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

78.2 Arguments

uplo	PLASMA_enum (IN) = PlasmaUpper: Upper triangle of A is stored; = PlasmaLower: Lower triangle of A is stored.
A	PLASMA_Complex64_t* (IN) The triangular factor U or L from the Cholesky factorization $A = U**H*U$ or $A = L*L$ ↵ **H, computed by PLASMA_zpotrf .
B	PLASMA_Complex64_t* (INOUT) On entry, the N-by-NRHS right hand side matrix B. On exit, if return value = 0, the N-by-NRHS solution matrix X.

78.3 Return Value:

= 0: successful exit

78.4 C Bindings

```
int PLASMA_zpotrs_Tile(PLASMA_enum uplo, PLASMA_desc *A, PLASMA_desc *B)
```

78.5 Fortran Bindings

```
void PLASMA_ZPOTRS_TILE(PLASMA_enum *uplo, long long int *A, long long int *B, int *INFO)
```

78.6 Online Browsing

Dive into [PLASMA_zpotrs_Tile](#)

79 PLASMA_ztrsm

79.1 Purpose

PLASMA_ztrsm - Computes triangular solve $A*X = B$ or $X*A = B$

79.2 Arguments

side	PLASMA_enum (IN) Specifies whether A appears on the left or on the right of X: = PlasmaLeft: $A*X = B$ = PlasmaRight: $X*A = B$
uplo	PLASMA_enum (IN) Specifies whether the matrix A is upper triangular or lower triangular: = PlasmaUpper: Upper triangle of A is stored; = PlasmaLower: Lower triangle of A is stored.
transA	PLASMA_enum (IN) Specifies whether the matrix A is transposed, not transposed or conjugate transposed: \leftrightarrow = PlasmaNoTrans: A is transposed; = PlasmaTrans: A is not transposed; = PlasmaConjTrans: A is conjugate transposed.
diag	PLASMA_enum (IN) Specifies whether or not A is unit triangular: = PlasmaNonUnit: A is non unit; = PlasmaUnit: A is unit.
N	int (IN) The order of the matrix A. $N \geq 0$.
NRHS	int (IN) The number of right hand sides, i.e., the number of columns of the matrix B. NRHS \leftrightarrow ≥ 0 .
A	PLASMA_Complex64_t* (IN) The triangular matrix A. If uplo = PlasmaUpper, the leading N-by-N upper triangular part of the array A contains the upper triangular matrix, and the strictly lower triangular part of A is not referenced. If uplo = PlasmaLower, the leading N-by-N lower triangular part of the array A contains the lower triangular matrix, and the strictly upper triangular part of A is not referenced. If diag = PlasmaUnit, the diagonal elements of A are also not referenced and are assumed to be 1.
LDA	int (IN) The leading dimension of the array A. $LDA \geq \max(1, N)$.
B	PLASMA_Complex64_t* (INOUT) On entry, the N-by-NRHS right hand side matrix B. On exit, if return value = 0, the N-by-NRHS solution matrix X.
LDB	PLASMA_Complex64_t* (IN)

The leading dimension of the array B. $LDB \geq \max(1, N)$.

79.3 Return Value:

= 0: successful exit
 < 0: if -i, the i-th argument had an illegal value

79.4 C Bindings

```
int PLASMA_ztrsm(PLASMA_enum side, PLASMA_enum uplo, PLASMA_enum transA, PLASMA_enum diag, ↵
    int N, int NRHS, PLASMA_Complex64_t *A, int LDA, PLASMA_Complex64_t *B, int LDB)
```

79.5 Fortran Bindings

```
void PLASMA_ZTRSM(PLASMA_enum *side, PLASMA_enum *uplo, PLASMA_enum *transA, PLASMA_enum * ↵
    diag, int *N, int *NRHS, PLASMA_Complex64_t *A, int *LDA, PLASMA_Complex64_t *B, int * ↵
    LDB, int *INFO)
```

79.6 Online Browsing

Dive into [PLASMA_ztrsm](#)

80 PLASMA_ztrsm_Tile

80.1 Purpose

PLASMA_ztrsm_Tile - Computes triangular solve $A*X = B$ or $X*A = B$ All matrices are passed through descriptors. All dimensions are taken from the descriptors.

80.2 Arguments

side	PLASMA_enum (IN) Specifies whether A appears on the left or on the right of X: = PlasmaLeft: $A*X = B$ = PlasmaRight: $X*A = B$
uplo	PLASMA_enum (IN) Specifies whether the matrix A is upper triangular or lower triangular: = PlasmaUpper: Upper triangle of A is stored; = PlasmaLower: Lower triangle of A is stored.
transA	PLASMA_enum (IN) Specifies whether the matrix A is transposed, not transposed or conjugate ↵ transposed: = PlasmaNoTrans: A is transposed; = PlasmaTrans: A is not transposed; = PlasmaConjTrans: A is conjugate transposed.
diag	PLASMA_enum (IN) Specifies whether or not A is unit triangular:

```

    = PlasmaNonUnit: A is non unit;
    = PlasmaUnit:    A us unit.

A    PLASMA_Complex64_t* (IN)
    The triangular matrix A. If uplo = PlasmaUpper, the leading N-by-N upper ↔
        triangular
    part of the array A contains the upper triangular matrix, and the strictly lower
    triangular part of A is not referenced. If uplo = PlasmaLower, the leading N-by-N
    lower triangular part of the array A contains the lower triangular matrix, and the
    strictly upper triangular part of A is not referenced. If diag = PlasmaUnit, the
    diagonal elements of A are also not referenced and are assumed to be 1.

B    PLASMA_Complex64_t* (INOUT)
    On entry, the N-by-NRHS right hand side matrix B.
    On exit, if return value = 0, the N-by-NRHS solution matrix X.

```

80.3 Return Value:

```

    = 0: successful exit

```

80.4 C Bindings

```

int PLASMA_ztrsm_Tile(PLASMA_enum side, PLASMA_enum uplo, PLASMA_enum transA, PLASMA_enum ↔
    diag, PLASMA_desc *A, PLASMA_desc *B)

```

80.5 Fortran Bindings

```

void PLASMA_ZTRSM_TILE(PLASMA_enum *side, PLASMA_enum *uplo, PLASMA_enum *transA, ↔
    PLASMA_enum *diag, long long int *A, long long int *B, int *INFO)

```

80.6 Online Browsing

Dive into [PLASMA_ztrsm_Tile](#)

81 PLASMA_ztrsmpi

81.1 Purpose

PLASMA_ztrsmpi - Performs the forward substitution step of solving a system of linear equations after the tile LU factorization of the matrix.

81.2 Arguments

```

N    int (IN)
    The order of the matrix A. N >= 0.

NRHS int (IN)
    The number of right hand sides, i.e., the number of columns of the matrix B. NRHS ↔
    >= 0.

```

A	PLASMA_Complex64_t* (IN) The tile factor L from the factorization, computed by PLASMA_zgetrf.
LDA	int (IN) The leading dimension of the array A. LDA >= max(1,N).
L	PLASMA_Complex64_t* (IN) Auxiliary factorization data, related to the tile L factor, computed by ↵ PLASMA_zgetrf.
IPIV	int* (IN) The pivot indices from PLASMA_zgetrf (not equivalent to LAPACK).
B	PLASMA_Complex64_t* (INOUT) On entry, the N-by-NRHS right hand side matrix B. On exit, if return value = 0, the N-by-NRHS solution matrix X.
LDB	PLASMA_Complex64_t* (IN) The leading dimension of the array B. LDB >= max(1,N).

81.3 Return Value:

```
= 0: successful exit
< 0: if -i, the i-th argument had an illegal value
```

81.4 C Bindings

```
int PLASMA_ztrsmpl(int N, int NRHS, PLASMA_Complex64_t *A, int LDA, PLASMA_Complex64_t *L, ↵
int *IPIV, PLASMA_Complex64_t *B, int LDB)
```

81.5 Fortran Bindings

```
void PLASMA_ZTRSMPL(int *N, int *NRHS, PLASMA_Complex64_t *A, int *LDA, PLASMA_Complex64_t ↵
**LH, int **IPIVH, PLASMA_Complex64_t *B, int *LDB, int *INFO)
```

81.6 Online Browsing

Dive into [PLASMA_ztrsmpl](#)

82 PLASMA_ztrsmpl_Tile

82.1 Purpose

PLASMA_ztrsmpl_Tile - Performs the forward substitution step of solving a system of linear equations after the tile LU factorization of the matrix. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

82.2 Arguments

A	PLASMA_Complex64_t* (IN) The tile factor L from the factorization, computed by PLASMA_zgetrf.
L	PLASMA_Complex64_t* (IN) Auxiliary factorization data, related to the tile L factor, computed by \leftrightarrow PLASMA_zgetrf.
IPIV	int* (IN) The pivot indices from PLASMA_zgetrf (not equivalent to LAPACK).
B	PLASMA_Complex64_t* (INOUT) On entry, the N-by-NRHS right hand side matrix B. On exit, if return value = 0, the N-by-NRHS solution matrix X.

82.3 Return Value:

= 0: successful exit

82.4 C Bindings

```
int PLASMA_ztrsmpl_Tile(PLASMA_desc *A, PLASMA_desc *L, int *IPIV, PLASMA_desc *B)
```

82.5 Fortran Bindings

```
void PLASMA_ZTRSMPL_TILE(long long int *A, long long int *L, int **IPIVH, long long int *B,  $\leftrightarrow$  int *INFO)
```

82.6 Online Browsing

Dive into [PLASMA_ztrsmpl_Tile](#)

83 PLASMA_zunglq

83.1 Purpose

PLASMA_zunglq - Generates an M-by-N matrix Q with orthonormal rows, which is defined as the first M rows of a product of the elementary reflectors returned by PLASMA_zgelqf.

83.2 Arguments

M	int (IN) The number of rows of the matrix Q. M >= 0.
N	int (IN) The number of columns of the matrix Q. N >= M.
K	int (IN)

	The number of rows of elementary tile reflectors whose product defines the matrix Q . $M \geq K \geq 0$.
A	PLASMA_Complex64_t* (IN) Details of the LQ factorization of the original matrix A as returned by PLASMA_zgelqf.
LDA	int (IN) The leading dimension of the array A. $LDA \geq \max(1, M)$.
T	PLASMA_Complex64_t* (IN) Auxiliary factorization data, computed by PLASMA_zgelqf.
B	PLASMA_Complex64_t* (OUT) On exit, the M-by-N matrix Q.
LDB	int (IN) The leading dimension of the array B. $LDB \geq \max(1, M)$.

83.3 Return Value:

```
= 0: successful exit
< 0: if -i, the i-th argument had an illegal value
```

83.4 C Bindings

```
int PLASMA_zunglq(int M, int N, int K, PLASMA_Complex64_t *A, int LDA, PLASMA_Complex64_t *T, PLASMA_Complex64_t *B, int LDB)
```

83.5 Fortran Bindings

```
void PLASMA_ZUNGLQ(int *M, int *N, int *K, PLASMA_Complex64_t *A, int *LDA, PLASMA_Complex64_t **T, PLASMA_Complex64_t *B, int *LDB, int *INFO)
```

83.6 Online Browsing

Dive into [PLASMA_zunglq](#)

84 PLASMA_zunglq_Tile

84.1 Purpose

PLASMA_zunglq_Tile - Generates an M-by-N matrix Q with orthonormal rows, which is defined as the first M rows of a product of the elementary reflectors returned by PLASMA_zgelqf_Tile. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

84.2 Arguments

A	PLASMA_Complex64_t* (IN) Details of the LQ factorization of the original matrix A as returned by <code>PLASMA_zgelqf</code> .
T	PLASMA_Complex64_t* (IN) Auxiliary factorization data, computed by <code>PLASMA_zgelqf</code> .
B	PLASMA_Complex64_t* (OUT) On exit, the M-by-N matrix Q.

84.3 Return Value:

= 0: successful exit

84.4 C Bindings

```
int PLASMA_zunglq_Tile(PLASMA_desc *A, PLASMA_desc *T, PLASMA_desc *B)
```

84.5 Fortran Bindings

84.6 Online Browsing

Dive into [PLASMA_zunglq_Tile](#)

85 PLASMA_zungqr

85.1 Purpose

PLASMA_zungqr - Generates an M-by-N matrix Q with orthonormal columns, which is defined as the first N columns of a product of the elementary reflectors returned by `PLASMA_zgeqrf`.

85.2 Arguments

M	<code>int</code> (IN) The number of rows of the matrix Q. $M \geq 0$.
N	<code>int</code> (IN) The number of columns of the matrix Q. $N \geq M$.
K	<code>int</code> (IN) The number of columns of elementary tile reflectors whose product defines the matrix Q. $M \geq K \geq 0$.
A	PLASMA_Complex64_t* (IN) Details of the QR factorization of the original matrix A as returned by <code>PLASMA_zgeqrf</code> .

```

LDA      int (IN)
          The leading dimension of the array A. LDA >= max(1,M).

T        PLASMA_Complex64_t* (IN)
          Auxiliary factorization data, computed by PLASMA_zgeqrf.

B        PLASMA_Complex64_t* (OUT)
          On exit, the M-by-N matrix Q.

LDB      int (IN)
          The leading dimension of the array B. LDB >= max(1,M).

```

85.3 Return Value:

```

= 0: successful exit
< 0: if -i, the i-th argument had an illegal value

```

85.4 C Bindings

```

int PLASMA_zungqr(int M, int N, int K, PLASMA_Complex64_t *A, int LDA, PLASMA_Complex64_t * ←
T, PLASMA_Complex64_t *B, int LDB)

```

85.5 Fortran Bindings

```

void PLASMA_ZUNGQR(int *M, int *N, int *K, PLASMA_Complex64_t *A, int *LDA, ←
PLASMA_Complex64_t **T, PLASMA_Complex64_t *B, int *LDB, int *INFO)

```

85.6 Online Browsing

Dive into [PLASMA_zungqr](#)

86 PLASMA_zungqr_Tile

86.1 Purpose

PLASMA_zungqr_Tile - Generates an M-by-N matrix Q with orthonormal columns, which is defined as the first N columns of a product of the elementary reflectors returned by **PLASMA_zgeqrf_Tile**. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

86.2 Arguments

```

A        PLASMA_Complex64_t* (IN)
          Details of the QR factorization of the original matrix A as returned by ←
          PLASMA_zgeqrf.

T        PLASMA_Complex64_t* (IN)
          Auxiliary factorization data, computed by PLASMA_zgeqrf.

B        PLASMA_Complex64_t* (OUT)
          On exit, the M-by-N matrix Q.

```

86.3 Return Value:

```
= 0: successful exit
```

86.4 C Bindings

```
int PLASMA_zungqr_Tile(PLASMA_desc *A, PLASMA_desc *T, PLASMA_desc *B)
```

86.5 Fortran Bindings

```
void PLASMA_ZUNGQR_TILE(long long int *A, long long int *T, long long int *B, int *INFO)
```

86.6 Online Browsing

Dive into [PLASMA_zungqr_Tile](#)

87 PLASMA_zunmlq

87.1 Purpose

PLASMA_zunmlq - overwrites the general M-by-N matrix C with Q^*C , where Q is an orthogonal matrix (unitary in the complex case) defined as the product of elementary reflectors returned by **PLASMA_zgelqf**. Q is of order M.

87.2 Arguments

side	PLASMA_enum (IN) Intended usage: = PlasmaLeft: apply Q or Q^*H from the left; = PlasmaRight: apply Q or Q^*H from the right. Currently only PlasmaLeft is supported.
trans	PLASMA_enum (IN) Intended usage: = PlasmaNoTrans: no transpose, apply Q; = PlasmaConjTrans: conjugate transpose, apply Q^*H . Currently only PlasmaConjTrans is supported.
M	<code>int</code> (IN) The number of rows of the matrix C. $M \geq 0$.
N	<code>int</code> (IN) The number of columns of the matrix C. $N \geq 0$.
K	<code>int</code> (IN) The number of rows of elementary tile reflectors whose product defines the matrix \leftrightarrow Q. $M \geq K \geq 0$.
A	PLASMA_Complex64_t* (IN) Details of the LQ factorization of the original matrix A as returned by \leftrightarrow PLASMA_zgelqf.

LDA	<code>int</code> (IN) The leading dimension of the array A. $LDA \geq \max(1, K)$.
T	<code>PLASMA_Complex64_t*</code> (IN) Auxiliary factorization data, computed by <code>PLASMA_zgelqf</code> .
B	<code>PLASMA_Complex64_t*</code> (INOUT) On entry, the M-by-N matrix B. On exit, B is overwritten by $Q*B$ or $Q**H*B$.
LDB	<code>int</code> (IN) The leading dimension of the array C. $LDB \geq \max(1, M)$.

87.3 Return Value:

```
= 0: successful exit
< 0: if -i, the i-th argument had an illegal value
```

87.4 C Bindings

```
int PLASMA_zunmlq(PLASMA_enum side, PLASMA_enum trans, int M, int N, int K, ↵
    PLASMA_Complex64_t *A, int LDA, PLASMA_Complex64_t *T, PLASMA_Complex64_t *B, int LDB)
```

87.5 Fortran Bindings

```
void PLASMA_ZUNMLQ(PLASMA_enum *side, PLASMA_enum *trans, int *M, int *N, int *K, ↵
    PLASMA_Complex64_t *A, int *LDA, PLASMA_Complex64_t **T, PLASMA_Complex64_t *B, int *LDB ↵
    , int *INFO)
```

87.6 Online Browsing

Dive into [PLASMA_zunmlq](#)

88 PLASMA_zunmlq_Tile

88.1 Purpose

PLASMA_zunmlq_Tile - overwrites the general M-by-N matrix C with $Q*C$, where Q is an orthogonal matrix (unitary in the complex case) defined as the product of elementary reflectors returned by `PLASMA_zgelqf_Tile` Q is of order M. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

88.2 Arguments

side	<code>PLASMA_enum</code> (IN) Intended usage: = PlasmaLeft: apply Q or $Q**H$ from the left; = PlasmaRight: apply Q or $Q**H$ from the right. Currently only PlasmaLeft is supported.
trans	<code>PLASMA_enum</code> (IN)

Intended usage:
 = PlasmaNoTrans: no transpose, apply Q;
 = PlasmaConjTrans: conjugate transpose, apply Q**H.
 Currently only PlasmaConjTrans is supported.

A PLASMA_Complex64_t* (IN)
 Details of the LQ factorization of the original matrix A as returned by `PLASMA_zgelqf`.

T PLASMA_Complex64_t* (IN)
 Auxiliary factorization data, computed by `PLASMA_zgelqf`.

B PLASMA_Complex64_t* (INOUT)
 On entry, the M-by-N matrix B.
 On exit, B is overwritten by Q*B or Q**H*B.

88.3 Return Value:

= 0: successful exit

88.4 C Bindings

```
int PLASMA_zunmlq_Tile(PLASMA_enum side, PLASMA_enum trans, PLASMA_desc *A, PLASMA_desc *T, ←
    PLASMA_desc *B)
```

88.5 Fortran Bindings

```
void PLASMA_ZUNMLQ_TILE(PLASMA_enum *side, PLASMA_enum *trans, long long int *A, long long ←
    int *T, long long int *B, int *INFO)
```

88.6 Online Browsing

Dive into [PLASMA_zunmlq_Tile](#)

89 PLASMA_zunmqr

89.1 Purpose

PLASMA_zunmqr - overwrites the general M-by-N matrix C with Q*C, where Q is an orthogonal matrix (unitary in the complex case) defined as the product of elementary reflectors returned by `PLASMA_zgeqrf`. Q is of order M.

89.2 Arguments

side PLASMA_enum (IN)
 Intended usage:
 = PlasmaLeft: apply Q or Q**H from the left;
 = PlasmaRight: apply Q or Q**H from the right.
 Currently only PlasmaLeft is supported.

trans PLASMA_enum (IN)

Intended usage:
 = PlasmaNoTrans: no transpose, apply Q;
 = PlasmaConjTrans: conjugate transpose, apply Q**H.
 Currently only PlasmaConjTrans is supported.

M **int** (IN)
 The number of rows of the matrix C. $M \geq 0$.

N **int** (IN)
 The number of columns of the matrix C. $N \geq 0$.

K **int** (IN)
 The number of columns of elementary tile reflectors whose product defines the \leftrightarrow matrix Q.
 $M \geq K \geq 0$.

A PLASMA_Complex64_t* (IN)
 Details of the QR factorization of the original matrix A as returned by \leftrightarrow PLASMA_zgeqrf.

LDA **int** (IN)
 The leading dimension of the array A. $LDA \geq \max(1, M)$;

T PLASMA_Complex64_t* (IN)
 Auxiliary factorization data, computed by PLASMA_zgeqrf.

B PLASMA_Complex64_t* (INOUT)
 On entry, the M-by-N matrix B.
 On exit, B is overwritten by $Q*B$ or $Q**H*B$.

LDB **int** (IN)
 The leading dimension of the array C. $LDB \geq \max(1, M)$.

89.3 Return Value:

= 0: successful exit
 < 0: **if** -i, the i-th argument had an illegal value

89.4 C Bindings

```
int PLASMA_zunmqr(PLASMA_enum side, PLASMA_enum trans, int M, int N, int K,  $\leftrightarrow$ 
    PLASMA_Complex64_t *A, int LDA, PLASMA_Complex64_t *T, PLASMA_Complex64_t *B, int LDB)
```

89.5 Fortran Bindings

```
void PLASMA_ZUNMQR(PLASMA_enum *side, PLASMA_enum *trans, int *M, int *N, int *K,  $\leftrightarrow$ 
    PLASMA_Complex64_t *A, int *LDA, PLASMA_Complex64_t **T, PLASMA_Complex64_t *B, int *LDB  $\leftrightarrow$ 
    , int *INFO)
```

89.6 Online Browsing

Dive into [PLASMA_zunmqr](#)

90 PLASMA_zunmqr_Tile

90.1 Purpose

PLASMA_zunmqr_Tile - overwrites the general M-by-N matrix C with Q^*C , where Q is an orthogonal matrix (unitary in the complex case) defined as the product of elementary reflectors returned by **PLASMA_zgeqrf_Tile**. Q is of order M. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

90.2 Arguments

side	PLASMA_enum (IN) Intended usage: = PlasmaLeft: apply Q or Q^{*H} from the left; = PlasmaRight: apply Q or Q^{*H} from the right. Currently only PlasmaLeft is supported.
trans	PLASMA_enum (IN) Intended usage: = PlasmaNoTrans: no transpose, apply Q; = PlasmaConjTrans: conjugate transpose, apply Q^{*H} . Currently only PlasmaConjTrans is supported.
A	PLASMA_Complex64_t* (IN) Details of the QR factorization of the original matrix A as returned by <code>PLASMA_zgeqrf</code> .
T	PLASMA_Complex64_t* (IN) Auxiliary factorization data, computed by <code>PLASMA_zgeqrf</code> .
B	PLASMA_Complex64_t* (INOUT) On entry, the M-by-N matrix B. On exit, B is overwritten by Q^*B or $Q^{*H}B$.

90.3 Return Value:

= 0: successful exit

90.4 C Bindings

```
int PLASMA_zunmqr_Tile(PLASMA_enum side, PLASMA_enum trans, PLASMA_desc *A, PLASMA_desc *T, ↵
    PLASMA_desc *B)
```

90.5 Fortran Bindings

```
void PLASMA_ZUNMQR_TILE(PLASMA_enum *side, PLASMA_enum *trans, long long int *A, long long ↵
    int *T, long long int *B, int *INFO)
```

90.6 Online Browsing

Dive into [PLASMA_zunmqr_Tile](#)