
COLLABORATORS

	<i>TITLE :</i>		
<i>ACTION</i>	<i>NAME</i>	<i>DATE</i>	<i>SIGNATURE</i>
WRITTEN BY		2009-11-15	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

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

6	PLASMA_cgels	5
6.1	Purpose	5
6.2	C Bindings	5
6.3	Fortran Bindings	5
6.4	Arguments	6
6.5	Return Value:	6
6.6	Online Browsing	6
7	PLASMA_cgels_Tile	7
7.1	Purpose	7
7.2	C Bindings	7
7.3	Fortran Bindings	7
7.4	Arguments	7
7.5	Return Value:	8
7.6	Online Browsing	8
8	PLASMA_cgemm	8
8.1	Purpose	8
8.2	C Bindings	8
8.3	Fortran Bindings	8
8.4	Arguments	8
8.5	Return Value:	9
8.6	Online Browsing	9
9	PLASMA_cgeqrf	9
9.1	Purpose	9
9.2	C Bindings	10
9.3	Fortran Bindings	10
9.4	Arguments	10
9.5	Return Value:	10
9.6	Online Browsing	10
10	PLASMA_cgeqrf_Tile	10
10.1	Purpose	10
10.2	C Bindings	11
10.3	Fortran Bindings	11
10.4	Arguments	11
10.5	Return Value:	11
10.6	Online Browsing	11

11 PLASMA_cgeqrs	11
11.1 Purpose	11
11.2 C Bindings	11
11.3 Fortran Bindings	11
11.4 Arguments	12
11.5 Return Value:	12
11.6 Online Browsing	12
12 PLASMA_cgeqrs_Tile	12
12.1 Purpose	12
12.2 C Bindings	12
12.3 Fortran Bindings	12
12.4 Arguments	13
12.5 Return Value:	13
12.6 Online Browsing	13
13 PLASMA_cgesv	13
13.1 Purpose	13
13.2 C Bindings	13
13.3 Fortran Bindings	13
13.4 Arguments	13
13.5 Return Value:	14
13.6 Online Browsing	14
14 PLASMA_cgesv_Tile	14
14.1 Purpose	14
14.2 C Bindings	14
14.3 Fortran Bindings	14
14.4 Arguments	15
14.5 Return Value:	15
14.6 Online Browsing	15
15 PLASMA_cgetrf	15
15.1 Purpose	15
15.2 C Bindings	15
15.3 Fortran Bindings	15
15.4 Arguments	16
15.5 Return Value:	16
15.6 Online Browsing	16

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

21 PLASMA_Set	20
21.1 Purpose	20
21.2 C Bindings	21
21.3 Fortran Bindings	21
21.4 Arguments	21
21.5 Return Value:	21
21.6 Online Browsing	21
22 PLASMA_Get	21
22.1 Purpose	21
22.2 C Bindings	21
22.3 Fortran Bindings	21
22.4 Arguments	21
22.5 Return Value:	22
22.6 Online Browsing	22
23 PLASMA_Init	22
23.1 Purpose	22
23.2 C Bindings	22
23.3 Fortran Bindings	22
23.4 Arguments	22
23.5 Return Value:	22
23.6 Online Browsing	22
24 PLASMA_Finalize	22
24.1 Purpose	22
24.2 C Bindings	22
24.3 Fortran Bindings	23
24.4 Online Browsing	23
25 PLASMA_cposv	23
25.1 Purpose	23
25.2 C Bindings	23
25.3 Fortran Bindings	23
25.4 Arguments	23
25.5 Return Value:	24
25.6 Online Browsing	24

26 PLASMA_cposv_Tile	24
26.1 Purpose	24
26.2 C Bindings	24
26.3 Fortran Bindings	24
26.4 Arguments	25
26.5 Return Value:	25
26.6 Online Browsing	25
27 PLASMA_cpotrf	25
27.1 Purpose	25
27.2 C Bindings	25
27.3 Fortran Bindings	26
27.4 Arguments	26
27.5 Return Value:	26
27.6 Online Browsing	26
28 PLASMA_cpotrf_Tile	26
28.1 Purpose	26
28.2 C Bindings	27
28.3 Fortran Bindings	27
28.4 Arguments	27
28.5 Return Value:	27
28.6 Online Browsing	27
29 PLASMA_cpotsr	27
29.1 Purpose	27
29.2 C Bindings	27
29.3 Fortran Bindings	28
29.4 Arguments	28
29.5 Return Value:	28
29.6 Online Browsing	28
30 PLASMA_cpotsr_Tile	28
30.1 Purpose	28
30.2 C Bindings	28
30.3 Fortran Bindings	29
30.4 Arguments	29
30.5 Return Value:	29
30.6 Online Browsing	29

31 PLASMA_ctrsm	29
31.1 Purpose	29
31.2 C Bindings	29
31.3 Fortran Bindings	29
31.4 Arguments	30
31.5 Return Value:	30
31.6 Online Browsing	30
32 PLASMA_ctrsm_Tile	31
32.1 Purpose	31
32.2 C Bindings	31
32.3 Fortran Bindings	31
32.4 Arguments	31
32.5 Return Value:	32
32.6 Online Browsing	32
33 PLASMA_ctrsmpl	32
33.1 Purpose	32
33.2 C Bindings	32
33.3 Fortran Bindings	32
33.4 Arguments	32
33.5 Return Value:	33
33.6 Online Browsing	33
34 PLASMA_ctrsmpl_Tile	33
34.1 Purpose	33
34.2 C Bindings	33
34.3 Fortran Bindings	33
34.4 Arguments	33
34.5 Return Value:	33
34.6 Online Browsing	33
35 PLASMA_cunglq	34
35.1 Purpose	34
35.2 C Bindings	34
35.3 Fortran Bindings	34
35.4 Arguments	34
35.5 Return Value:	34
35.6 Online Browsing	35

36 PLASMA_cunglq_Tile	35
36.1 Purpose	35
36.2 C Bindings	35
36.3 Fortran Bindings	35
36.4 Arguments	35
36.5 Return Value:	35
36.6 Online Browsing	35
37 PLASMA_cungqr	35
37.1 Purpose	35
37.2 C Bindings	36
37.3 Fortran Bindings	36
37.4 Arguments	36
37.5 Return Value:	36
37.6 Online Browsing	36
38 PLASMA_cungqr_Tile	37
38.1 Purpose	37
38.2 C Bindings	37
38.3 Fortran Bindings	37
38.4 Arguments	37
38.5 Return Value:	37
38.6 Online Browsing	37
39 PLASMA_cunmlq	37
39.1 Purpose	37
39.2 C Bindings	37
39.3 Fortran Bindings	38
39.4 Arguments	38
39.5 Return Value:	38
39.6 Online Browsing	38
40 PLASMA_cunmlq_Tile	39
40.1 Purpose	39
40.2 C Bindings	39
40.3 Fortran Bindings	39
40.4 Arguments	39
40.5 Return Value:	39
40.6 Online Browsing	39

41 PLASMA_cunmqr	40
41.1 Purpose	40
41.2 C Bindings	40
41.3 Fortran Bindings	40
41.4 Arguments	40
41.5 Return Value:	41
41.6 Online Browsing	41
42 PLASMA_cunmqr_Tile	41
42.1 Purpose	41
42.2 C Bindings	41
42.3 Fortran Bindings	41
42.4 Arguments	41
42.5 Return Value:	42
42.6 Online Browsing	42
43 PLASMA_Desc_Create	42
43.1 Purpose	42
43.2 C Bindings	42
43.3 Fortran Bindings	42
43.4 Arguments	42
43.5 Return Value:	43
43.6 Online Browsing	43
44 PLASMA_Desc_Destroy	43
44.1 Purpose	43
44.2 C Bindings	43
44.3 Fortran Bindings	43
44.4 Arguments	43
44.5 Return Value:	44
44.6 Online Browsing	44
45 PLASMA_dgelqf	44
45.1 Purpose	44
45.2 C Bindings	44
45.3 Fortran Bindings	44
45.4 Arguments	44
45.5 Return Value:	45
45.6 Online Browsing	45

46 PLASMA_dgelqf_Tile	45
46.1 Purpose	45
46.2 C Bindings	45
46.3 Fortran Bindings	45
46.4 Arguments	45
46.5 Return Value:	45
46.6 Online Browsing	45
47 PLASMA_dgelqs	46
47.1 Purpose	46
47.2 C Bindings	46
47.3 Fortran Bindings	46
47.4 Arguments	46
47.5 Return Value:	46
47.6 Online Browsing	47
48 PLASMA_dgelqs_Tile	47
48.1 Purpose	47
48.2 C Bindings	47
48.3 Fortran Bindings	47
48.4 Arguments	47
48.5 Return Value:	47
48.6 Online Browsing	47
49 PLASMA_dgels	47
49.1 Purpose	47
49.2 C Bindings	48
49.3 Fortran Bindings	48
49.4 Arguments	48
49.5 Return Value:	49
49.6 Online Browsing	49
50 PLASMA_dgels_Tile	49
50.1 Purpose	49
50.2 C Bindings	49
50.3 Fortran Bindings	49
50.4 Arguments	49
50.5 Return Value:	50
50.6 Online Browsing	50

51 PLASMA_dgemm	50
51.1 Purpose	50
51.2 C Bindings	50
51.3 Fortran Bindings	51
51.4 Arguments	51
51.5 Return Value:	52
51.6 Online Browsing	52
52 PLASMA_dgeqrf	52
52.1 Purpose	52
52.2 C Bindings	52
52.3 Fortran Bindings	52
52.4 Arguments	52
52.5 Return Value:	53
52.6 Online Browsing	53
53 PLASMA_dgeqrf_Tile	53
53.1 Purpose	53
53.2 C Bindings	53
53.3 Fortran Bindings	53
53.4 Arguments	53
53.5 Return Value:	53
53.6 Online Browsing	53
54 PLASMA_dgeqrs	54
54.1 Purpose	54
54.2 C Bindings	54
54.3 Fortran Bindings	54
54.4 Arguments	54
54.5 Return Value:	54
54.6 Online Browsing	55
55 PLASMA_dgeqrs_Tile	55
55.1 Purpose	55
55.2 C Bindings	55
55.3 Fortran Bindings	55
55.4 Arguments	55
55.5 Return Value:	55
55.6 Online Browsing	55

56	PLASMA_dgesv	55
56.1	Purpose	55
56.2	C Bindings	56
56.3	Fortran Bindings	56
56.4	Arguments	56
56.5	Return Value:	56
56.6	Online Browsing	56
57	PLASMA_dgesv_Tile	57
57.1	Purpose	57
57.2	C Bindings	57
57.3	Fortran Bindings	57
57.4	Arguments	57
57.5	Return Value:	57
57.6	Online Browsing	57
58	PLASMA_dgetrf	58
58.1	Purpose	58
58.2	C Bindings	58
58.3	Fortran Bindings	58
58.4	Arguments	58
58.5	Return Value:	58
58.6	Online Browsing	58
59	PLASMA_dgetrf_Tile	59
59.1	Purpose	59
59.2	C Bindings	59
59.3	Fortran Bindings	59
59.4	Arguments	59
59.5	Return Value:	59
59.6	Online Browsing	59
60	PLASMA_dgetrs	59
60.1	Purpose	59
60.2	C Bindings	60
60.3	Fortran Bindings	60
60.4	Arguments	60
60.5	Return Value:	60
60.6	Online Browsing	60

61 PLASMA_dgetrs_Tile	61
61.1 Purpose	61
61.2 C Bindings	61
61.3 Fortran Bindings	61
61.4 Arguments	61
61.5 Return Value:	61
61.6 Online Browsing	61
62 PLASMA_dposv	61
62.1 Purpose	61
62.2 C Bindings	62
62.3 Fortran Bindings	62
62.4 Arguments	62
62.5 Return Value:	63
62.6 Online Browsing	63
63 PLASMA_dposv_Tile	63
63.1 Purpose	63
63.2 C Bindings	63
63.3 Fortran Bindings	63
63.4 Arguments	63
63.5 Return Value:	64
63.6 Online Browsing	64
64 PLASMA_dpotrf	64
64.1 Purpose	64
64.2 C Bindings	64
64.3 Fortran Bindings	64
64.4 Arguments	64
64.5 Return Value:	65
64.6 Online Browsing	65
65 PLASMA_dpotrf_Tile	65
65.1 Purpose	65
65.2 C Bindings	65
65.3 Fortran Bindings	65
65.4 Arguments	66
65.5 Return Value:	66
65.6 Online Browsing	66

66 PLASMA_dpotrs	66
66.1 Purpose	66
66.2 C Bindings	66
66.3 Fortran Bindings	66
66.4 Arguments	67
66.5 Return Value:	67
66.6 Online Browsing	67
67 PLASMA_dpotrs_Tile	67
67.1 Purpose	67
67.2 C Bindings	67
67.3 Fortran Bindings	67
67.4 Arguments	68
67.5 Return Value:	68
67.6 Online Browsing	68
68 PLASMA_dsgesv	68
68.1 Purpose	68
68.2 C Bindings	69
68.3 Fortran Bindings	69
68.4 Arguments	69
68.5 Return Value:	69
68.6 Online Browsing	69
69 PLASMA_dsgesv_Tile	70
69.1 Purpose	70
69.2 C Bindings	70
69.3 Fortran Bindings	70
69.4 Arguments	70
69.5 Return Value:	71
69.6 Online Browsing	71
70 PLASMA_dtrsm	71
70.1 Purpose	71
70.2 C Bindings	71
70.3 Fortran Bindings	71
70.4 Arguments	71
70.5 Return Value:	72
70.6 Online Browsing	72

71 PLASMA_dtrsm_Tile	72
71.1 Purpose	72
71.2 C Bindings	73
71.3 Fortran Bindings	73
71.4 Arguments	73
71.5 Return Value:	73
71.6 Online Browsing	73
72 PLASMA_dtrsmpi	74
72.1 Purpose	74
72.2 C Bindings	74
72.3 Fortran Bindings	74
72.4 Arguments	74
72.5 Return Value:	74
72.6 Online Browsing	75
73 PLASMA_dtrsmpi_Tile	75
73.1 Purpose	75
73.2 C Bindings	75
73.3 Fortran Bindings	75
73.4 Arguments	75
73.5 Return Value:	75
73.6 Online Browsing	75
74 PLASMA_sgelqf	75
74.1 Purpose	75
74.2 C Bindings	76
74.3 Fortran Bindings	76
74.4 Arguments	76
74.5 Return Value:	76
74.6 Online Browsing	76
75 PLASMA_sgelqf_Tile	76
75.1 Purpose	76
75.2 C Bindings	77
75.3 Fortran Bindings	77
75.4 Arguments	77
75.5 Return Value:	77
75.6 Online Browsing	77

76 PLASMA_sgelqs	77
76.1 Purpose	77
76.2 C Bindings	77
76.3 Fortran Bindings	77
76.4 Arguments	78
76.5 Return Value:	78
76.6 Online Browsing	78
77 PLASMA_sgelqs_Tile	78
77.1 Purpose	78
77.2 C Bindings	78
77.3 Fortran Bindings	78
77.4 Arguments	79
77.5 Return Value:	79
77.6 Online Browsing	79
78 PLASMA_sgels	79
78.1 Purpose	79
78.2 C Bindings	79
78.3 Fortran Bindings	79
78.4 Arguments	80
78.5 Return Value:	80
78.6 Online Browsing	80
79 PLASMA_sgels_Tile	81
79.1 Purpose	81
79.2 C Bindings	81
79.3 Fortran Bindings	81
79.4 Arguments	81
79.5 Return Value:	82
79.6 Online Browsing	82
80 PLASMA_sgemm	82
80.1 Purpose	82
80.2 C Bindings	82
80.3 Fortran Bindings	82
80.4 Arguments	82
80.5 Return Value:	83
80.6 Online Browsing	83

81 PLASMA_sgeqrf	83
81.1 Purpose	83
81.2 C Bindings	84
81.3 Fortran Bindings	84
81.4 Arguments	84
81.5 Return Value:	84
81.6 Online Browsing	84
82 PLASMA_sgeqrf_Tile	84
82.1 Purpose	84
82.2 C Bindings	85
82.3 Fortran Bindings	85
82.4 Arguments	85
82.5 Return Value:	85
82.6 Online Browsing	85
83 PLASMA_sgeqrs	85
83.1 Purpose	85
83.2 C Bindings	85
83.3 Fortran Bindings	85
83.4 Arguments	86
83.5 Return Value:	86
83.6 Online Browsing	86
84 PLASMA_sgeqrs_Tile	86
84.1 Purpose	86
84.2 C Bindings	86
84.3 Fortran Bindings	86
84.4 Arguments	87
84.5 Return Value:	87
84.6 Online Browsing	87
85 PLASMA_sgesv	87
85.1 Purpose	87
85.2 C Bindings	87
85.3 Fortran Bindings	87
85.4 Arguments	87
85.5 Return Value:	88
85.6 Online Browsing	88

86 PLASMA_sgesv_Tile	88
86.1 Purpose	88
86.2 C Bindings	88
86.3 Fortran Bindings	88
86.4 Arguments	89
86.5 Return Value:	89
86.6 Online Browsing	89
87 PLASMA_sgetrf	89
87.1 Purpose	89
87.2 C Bindings	89
87.3 Fortran Bindings	89
87.4 Arguments	90
87.5 Return Value:	90
87.6 Online Browsing	90
88 PLASMA_sgetrf_Tile	90
88.1 Purpose	90
88.2 C Bindings	90
88.3 Fortran Bindings	90
88.4 Arguments	91
88.5 Return Value:	91
88.6 Online Browsing	91
89 PLASMA_sgetrs	91
89.1 Purpose	91
89.2 C Bindings	91
89.3 Fortran Bindings	91
89.4 Arguments	91
89.5 Return Value:	92
89.6 Online Browsing	92
90 PLASMA_sgetrs_Tile	92
90.1 Purpose	92
90.2 C Bindings	92
90.3 Fortran Bindings	92
90.4 Arguments	93
90.5 Return Value:	93
90.6 Online Browsing	93

91 PLASMA_sposv	93
91.1 Purpose	93
91.2 C Bindings	93
91.3 Fortran Bindings	93
91.4 Arguments	94
91.5 Return Value:	94
91.6 Online Browsing	94
92 PLASMA_sposv_Tile	94
92.1 Purpose	94
92.2 C Bindings	95
92.3 Fortran Bindings	95
92.4 Arguments	95
92.5 Return Value:	95
92.6 Online Browsing	95
93 PLASMA_spotrf	96
93.1 Purpose	96
93.2 C Bindings	96
93.3 Fortran Bindings	96
93.4 Arguments	96
93.5 Return Value:	96
93.6 Online Browsing	97
94 PLASMA_spotrf_Tile	97
94.1 Purpose	97
94.2 C Bindings	97
94.3 Fortran Bindings	97
94.4 Arguments	97
94.5 Return Value:	97
94.6 Online Browsing	98
95 PLASMA_spotrs	98
95.1 Purpose	98
95.2 C Bindings	98
95.3 Fortran Bindings	98
95.4 Arguments	98
95.5 Return Value:	98
95.6 Online Browsing	99

96 PLASMA_spotrs_Tile	99
96.1 Purpose	99
96.2 C Bindings	99
96.3 Fortran Bindings	99
96.4 Arguments	99
96.5 Return Value:	99
96.6 Online Browsing	99
97 PLASMA_strsm	99
97.1 Purpose	99
97.2 C Bindings	100
97.3 Fortran Bindings	100
97.4 Arguments	100
97.5 Return Value:	101
97.6 Online Browsing	101
98 PLASMA_strsm_Tile	101
98.1 Purpose	101
98.2 C Bindings	101
98.3 Fortran Bindings	101
98.4 Arguments	101
98.5 Return Value:	102
98.6 Online Browsing	102
99 PLASMA_strsmpl	102
99.1 Purpose	102
99.2 C Bindings	102
99.3 Fortran Bindings	102
99.4 Arguments	102
99.5 Return Value:	103
99.6 Online Browsing	103
100 PLASMA_strsmpl_Tile	103
100.1 Purpose	103
100.2 C Bindings	103
100.3 Fortran Bindings	103
100.4 Arguments	103
100.5 Return Value:	104
100.6 Online Browsing	104

101PLASMA_Lapack_to_Tile	104
101.1Purpose	104
101.2C Bindings	104
101.3Fortran Bindings	104
101.4Arguments	104
101.5Return Value:	104
101.6Online Browsing	105
102PLASMA_Tile_to_Lapack	105
102.1Purpose	105
102.2C Bindings	105
102.3Fortran Bindings	105
102.4Arguments	105
102.5Return Value:	105
102.6Online Browsing	105
103PLASMA_Dealloc_Handle	105
103.1Purpose	105
103.2C Bindings	105
103.3Fortran Bindings	106
103.4Arguments	106
103.5Return Value:	106
103.6Online Browsing	106
104PLASMA_Dealloc_Handle	106
104.1Purpose	106
104.2C Bindings	106
104.3Fortran Bindings	106
104.4Arguments	106
104.5Return Value:	106
104.6Online Browsing	106
105PLASMA_Alloc_Workspace_cgels	107
105.1Purpose	107
105.2C Bindings	107
105.3Fortran Bindings	107
105.4Arguments	107
105.5Return Value:	107
105.6Online Browsing	107

106	PLASMA_Alloc_Workspace_cgeqrf	107
106.1	Purpose	107
106.2	C Bindings	107
106.3	Fortran Bindings	108
106.4	Arguments	108
106.5	Return Value:	108
106.6	Online Browsing	108
107	PLASMA_Alloc_Workspace_cgels	108
107.1	Purpose	108
107.2	C Bindings	108
107.3	Fortran Bindings	108
107.4	Arguments	108
107.5	Return Value:	109
107.6	Online Browsing	109
108	PLASMA_Alloc_Workspace_cgesv	109
108.1	Purpose	109
108.2	C Bindings	109
108.3	Fortran Bindings	109
108.4	Arguments	109
108.5	Return Value:	109
108.6	Online Browsing	109
109	PLASMA_Alloc_Workspace_cgetrf	110
109.1	Purpose	110
109.2	C Bindings	110
109.3	Fortran Bindings	110
109.4	Arguments	110
109.5	Return Value:	110
109.6	Online Browsing	110
110	PLASMA_Alloc_Workspace_dgels	110
110.1	Purpose	110
110.2	C Bindings	110
110.3	Fortran Bindings	111
110.4	Arguments	111
110.5	Return Value:	111
110.6	Online Browsing	111

111	PLASMA_Alloc_Workspace_dgeqrf	111
111.1	Purpose	111
111.2	C Bindings	111
111.3	Fortran Bindings	111
111.4	Arguments	111
111.5	Return Value:	112
111.6	Online Browsing	112
112	PLASMA_Alloc_Workspace_dgelqf	112
112.1	Purpose	112
112.2	C Bindings	112
112.3	Fortran Bindings	112
112.4	Arguments	112
112.5	Return Value:	112
112.6	Online Browsing	112
113	PLASMA_Alloc_Workspace_dgesv	113
113.1	Purpose	113
113.2	C Bindings	113
113.3	Fortran Bindings	113
113.4	Arguments	113
113.5	Return Value:	113
113.6	Online Browsing	113
114	PLASMA_Alloc_Workspace_dgetrf	113
114.1	Purpose	113
114.2	C Bindings	113
114.3	Fortran Bindings	114
114.4	Arguments	114
114.5	Return Value:	114
114.6	Online Browsing	114
115	PLASMA_Alloc_Workspace_sgels	114
115.1	Purpose	114
115.2	C Bindings	114
115.3	Fortran Bindings	114
115.4	Arguments	115
115.5	Return Value:	115
115.6	Online Browsing	115

116	PLASMA_Alloc_Workspace_sgeqrf	115
116.1	Purpose	115
116.2	C Bindings	115
116.3	Fortran Bindings	115
116.4	Arguments	115
116.5	Return Value:	116
116.6	Online Browsing	116
117	PLASMA_Alloc_Workspace_sgelqf	116
117.1	Purpose	116
117.2	C Bindings	116
117.3	Fortran Bindings	116
117.4	Arguments	116
117.5	Return Value:	116
117.6	Online Browsing	116
118	PLASMA_Alloc_Workspace_sgesv	116
118.1	Purpose	116
118.2	C Bindings	117
118.3	Fortran Bindings	117
118.4	Arguments	117
118.5	Return Value:	117
118.6	Online Browsing	117
119	PLASMA_Alloc_Workspace_sgetrf	117
119.1	Purpose	117
119.2	C Bindings	117
119.3	Fortran Bindings	117
119.4	Arguments	118
119.5	Return Value:	118
119.6	Online Browsing	118
120	PLASMA_Alloc_Workspace_zgels	118
120.1	Purpose	118
120.2	C Bindings	118
120.3	Fortran Bindings	118
120.4	Arguments	118
120.5	Return Value:	119
120.6	Online Browsing	119

121	PLASMA_Alloc_Workspace_zgeqrf	119
121.1	Purpose	119
121.2	C Bindings	119
121.3	Fortran Bindings	119
121.4	Arguments	119
121.5	Return Value:	119
121.6	Online Browsing	119
122	PLASMA_Alloc_Workspace_zgelqf	119
122.1	Purpose	119
122.2	C Bindings	120
122.3	Fortran Bindings	120
122.4	Arguments	120
122.5	Return Value:	120
122.6	Online Browsing	120
123	PLASMA_Alloc_Workspace_zgesv	120
123.1	Purpose	120
123.2	C Bindings	120
123.3	Fortran Bindings	120
123.4	Arguments	121
123.5	Return Value:	121
123.6	Online Browsing	121
124	PLASMA_Alloc_Workspace_zgetrf	121
124.1	Purpose	121
124.2	C Bindings	121
124.3	Fortran Bindings	121
124.4	Arguments	121
124.5	Return Value:	122
124.6	Online Browsing	122
125	PLASMA_zcgesv	122
125.1	Purpose	122
125.2	C Bindings	122
125.3	Fortran Bindings	122
125.4	Arguments	123
125.5	Return Value:	123
125.6	Online Browsing	123

126PLASMA_zcgesv_Tile	123
126.1 Purpose	123
126.2 C Bindings	124
126.3 Fortran Bindings	124
126.4 Arguments	124
126.5 Return Value:	124
126.6 Online Browsing	125
127PLASMA_zgelqf	125
127.1 Purpose	125
127.2 C Bindings	125
127.3 Fortran Bindings	125
127.4 Arguments	125
127.5 Return Value:	125
127.6 Online Browsing	126
128PLASMA_zgelqf_Tile	126
128.1 Purpose	126
128.2 C Bindings	126
128.3 Fortran Bindings	126
128.4 Arguments	126
128.5 Return Value:	126
128.6 Online Browsing	126
129PLASMA_zgelqs	126
129.1 Purpose	126
129.2 C Bindings	127
129.3 Fortran Bindings	127
129.4 Arguments	127
129.5 Return Value:	127
129.6 Online Browsing	127
130PLASMA_zgelqs_Tile	127
130.1 Purpose	127
130.2 C Bindings	128
130.3 Fortran Bindings	128
130.4 Arguments	128
130.5 Return Value:	128
130.6 Online Browsing	128

131PLASMA_zgels	128
131.1Purpose	128
131.2C Bindings	128
131.3Fortran Bindings	129
131.4Arguments	129
131.5Return Value:	129
131.6Online Browsing	130
132PLASMA_zgels_Tile	130
132.1Purpose	130
132.2C Bindings	130
132.3Fortran Bindings	130
132.4Arguments	130
132.5Return Value:	131
132.6Online Browsing	131
133PLASMA_zgemm	131
133.1Purpose	131
133.2C Bindings	131
133.3Fortran Bindings	131
133.4Arguments	132
133.5Return Value:	132
133.6Online Browsing	133
134PLASMA_zgeqrf	133
134.1Purpose	133
134.2C Bindings	133
134.3Fortran Bindings	133
134.4Arguments	133
134.5Return Value:	133
134.6Online Browsing	134
135PLASMA_zgeqrf_Tile	134
135.1Purpose	134
135.2C Bindings	134
135.3Fortran Bindings	134
135.4Arguments	134
135.5Return Value:	134
135.6Online Browsing	134

136PLASMA_zgeqrs	134
136.1Purpose	134
136.2C Bindings	135
136.3Fortran Bindings	135
136.4Arguments	135
136.5Return Value:	135
136.6Online Browsing	135
 137PLASMA_zgeqrs_Tile	 135
137.1Purpose	135
137.2C Bindings	136
137.3Fortran Bindings	136
137.4Arguments	136
137.5Return Value:	136
137.6Online Browsing	136
 138PLASMA_zgesv	 136
138.1Purpose	136
138.2C Bindings	136
138.3Fortran Bindings	136
138.4Arguments	137
138.5Return Value:	137
138.6Online Browsing	137
 139PLASMA_zgesv_Tile	 137
139.1Purpose	137
139.2C Bindings	138
139.3Fortran Bindings	138
139.4Arguments	138
139.5Return Value:	138
139.6Online Browsing	138
 140PLASMA_zgetrf	 138
140.1Purpose	138
140.2C Bindings	138
140.3Fortran Bindings	139
140.4Arguments	139
140.5Return Value:	139
140.6Online Browsing	139

141PLASMA_zgetrf_Tile	139
141.1Purpose	139
141.2C Bindings	139
141.3Fortran Bindings	140
141.4Arguments	140
141.5Return Value:	140
141.6Online Browsing	140
142PLASMA_zgetrs	140
142.1Purpose	140
142.2C Bindings	140
142.3Fortran Bindings	140
142.4Arguments	141
142.5Return Value:	141
142.6Online Browsing	141
143PLASMA_zgetrs_Tile	141
143.1Purpose	141
143.2C Bindings	142
143.3Fortran Bindings	142
143.4Arguments	142
143.5Return Value:	142
143.6Online Browsing	142
144PLASMA_zposv	142
144.1Purpose	142
144.2C Bindings	142
144.3Fortran Bindings	143
144.4Arguments	143
144.5Return Value:	143
144.6Online Browsing	143
145PLASMA_zposv_Tile	144
145.1Purpose	144
145.2C Bindings	144
145.3Fortran Bindings	144
145.4Arguments	144
145.5Return Value:	144
145.6Online Browsing	145

146PLASMA_zpotrf	145
146.1Purpose	145
146.2C Bindings	145
146.3Fortran Bindings	145
146.4Arguments	145
146.5Return Value:	146
146.6Online Browsing	146
147PLASMA_zpotrf_Tile	146
147.1Purpose	146
147.2C Bindings	146
147.3Fortran Bindings	146
147.4Arguments	146
147.5Return Value:	147
147.6Online Browsing	147
148PLASMA_zpotrs	147
148.1Purpose	147
148.2C Bindings	147
148.3Fortran Bindings	147
148.4Arguments	147
148.5Return Value:	148
148.6Online Browsing	148
149PLASMA_zpotrs_Tile	148
149.1Purpose	148
149.2C Bindings	148
149.3Fortran Bindings	148
149.4Arguments	148
149.5Return Value:	148
149.6Online Browsing	149
150PLASMA_ztrsm	149
150.1Purpose	149
150.2C Bindings	149
150.3Fortran Bindings	149
150.4Arguments	149
150.5Return Value:	150
150.6Online Browsing	150

151PLASMA_ztrsm_Tile	150
151.1Purpose	150
151.2C Bindings	150
151.3Fortran Bindings	150
151.4Arguments	150
151.5Return Value:	151
151.6Online Browsing	151
152PLASMA_ztrsml	151
152.1Purpose	151
152.2C Bindings	151
152.3Fortran Bindings	151
152.4Arguments	152
152.5Return Value:	152
152.6Online Browsing	152
153PLASMA_ztrsml_Tile	152
153.1Purpose	152
153.2C Bindings	152
153.3Fortran Bindings	152
153.4Arguments	153
153.5Return Value:	153
153.6Online Browsing	153
154PLASMA_zunglq	153
154.1Purpose	153
154.2C Bindings	153
154.3Fortran Bindings	153
154.4Arguments	153
154.5Return Value:	154
154.6Online Browsing	154
155PLASMA_zunglq_Tile	154
155.1Purpose	154
155.2C Bindings	154
155.3Fortran Bindings	154
155.4Arguments	155
155.5Return Value:	155
155.6Online Browsing	155

156	PLASMA_zungqr	155
156.1	Purpose	155
156.2	C Bindings	155
156.3	Fortran Bindings	155
156.4	Arguments	155
156.5	Return Value:	156
156.6	Online Browsing	156
157	PLASMA_zungqr_Tile	156
157.1	Purpose	156
157.2	C Bindings	156
157.3	Fortran Bindings	156
157.4	Arguments	156
157.5	Return Value:	157
157.6	Online Browsing	157
158	PLASMA_zunmlq	157
158.1	Purpose	157
158.2	C Bindings	157
158.3	Fortran Bindings	157
158.4	Arguments	157
158.5	Return Value:	158
158.6	Online Browsing	158
159	PLASMA_zunmlq_Tile	158
159.1	Purpose	158
159.2	C Bindings	158
159.3	Fortran Bindings	158
159.4	Arguments	159
159.5	Return Value:	159
159.6	Online Browsing	159
160	PLASMA_zunmqr	159
160.1	Purpose	159
160.2	C Bindings	159
160.3	Fortran Bindings	159
160.4	Arguments	160
160.5	Return Value:	160
160.6	Online Browsing	160

161	PLASMA_zunmqr_Tile	161
161.1	Purpose	161
161.2	C Bindings	161
161.3	Fortran Bindings	161
161.4	Arguments	161
161.5	Return Value:	161
161.6	Online Browsing	161

1 PLASMA_Version

1.1 Purpose

PLASMA_Version - Reports PLASMA version number.

1.2 C Bindings

```
int PLASMA_Version(int *ver_major, int *ver_minor, int *ver_micro)
```

1.3 Fortran Bindings

```
PLASMA_VERSION(INTEGER VER_MAJOR, INTEGER VER_MINOR, INTEGER VER_MICRO, INTEGER INFO)
```

1.4 Arguments

ver_major	int (OUT) PLASMA major version number.
ver_minor	int (OUT) PLASMA minor version number.
ver_micro	int (OUT) PLASMA micro version number.

1.5 Return Value:

```
= PLASMA_SUCCESS: successful exit
```

1.6 Online Browsing

Dive into [PLASMA_Version](#)

2 PLASMA_cgelsqf

2.1 Purpose

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

2.2 C Bindings

```
int PLASMA_cgelsqf(int M, int N, PLASMA_Complex32_t *A, int LDA, PLASMA_Complex32_t *T)
```

2.3 Fortran Bindings

```
PLASMA_CGELSQF(INTEGER M, INTEGER N, COMPLEX A, INTEGER LDA, INTEGER T, INTEGER INFO)
```

2.4 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_Complex32_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 <code>if</code> $M \leq N$); 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_Complex32_t*</code> (OUT) On exit, auxiliary factorization data, required by <code>PLASMA_cgels</code> to solve the system of equations.

2.5 Return Value:

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

2.6 Online Browsing

Dive into [PLASMA_cgels](#)

3 PLASMA_cgels_Tile

3.1 Purpose

PLASMA_cgels_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.

3.2 C Bindings

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

3.3 Fortran Bindings

```
PLASMA_CGELS_TILE(INTEGER*4 A, INTEGER*4 T, INTEGER INFO)
```

3.4 Arguments

A PLASMA_Complex32_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(\leftarrow M,N) lower trapezoidal matrix L (L is lower triangular if $M \leq N$); the elements above \leftarrow the diagonal represent the unitary matrix Q as a product of elementary reflectors, \leftarrow stored by tiles.

T PLASMA_Complex32_t* (OUT)
 On exit, auxiliary factorization data, required by PLASMA_cgels to solve the \leftarrow system of equations.

3.5 Return Value:

= 0: successful exit

3.6 Online Browsing

Dive into [PLASMA_cgels_Tile](#)

4 PLASMA_cgels

4.1 Purpose

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

4.2 C Bindings

```
int PLASMA_cgels(int M, int N, int NRHS, PLASMA_Complex32_t *A, int LDA,  $\leftarrow$ 
    PLASMA_Complex32_t *T, PLASMA_Complex32_t *B, int LDB)
```

4.3 Fortran Bindings

```
PLASMA_CGELS(INTEGER M, INTEGER N, INTEGER NRHS, COMPLEX A, INTEGER LDA, INTEGER T,  $\leftarrow$ 
    COMPLEX B, INTEGER LDB, INTEGER INFO)
```

4.4 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	<code>int</code> (IN) The number of columns of B. NRHS >= 0.
A	<code>PLASMA_Complex32_t*</code> (IN) Details of the LQ factorization of the original matrix A as returned by <code>↔ PLASMA_cgelsf</code> .
LDA	<code>int</code> (IN) The leading dimension of the array A. LDA >= M.
T	<code>PLASMA_Complex32_t*</code> (IN) Auxiliary factorization data, computed by <code>PLASMA_cgelsf</code> .
B	<code>PLASMA_Complex32_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 >= N.

4.5 Return Value:

```

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

```

4.6 Online Browsing

Dive into [PLASMA_cgelsf](#)

5 PLASMA_cgelsf_Tile

5.1 Purpose

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

5.2 C Bindings

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

5.3 Fortran Bindings

```
PLASMA_CGELSFS_TILE(INTEGER*4 A, INTEGER*4 B, INTEGER*4 T, INTEGER INFO)
```

5.4 Arguments

A	PLASMA_Complex32_t* (IN) Details of the LQ factorization of the original matrix A as returned by <code>PLASMA_cgelsqf</code> .
T	PLASMA_Complex32_t* (IN) Auxiliary factorization data, computed by <code>PLASMA_cgelsqf</code> .
B	PLASMA_Complex32_t* (INOUT) On entry, the M-by-NRHS right hand side matrix B. On exit, the N-by-NRHS solution matrix X.

5.5 Return Value:

= 0: successful exit

5.6 Online Browsing

Dive into [PLASMA_cgelsqf_Tile](#)

6 PLASMA_cgels

6.1 Purpose

PLASMA_cgels - 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.

6.2 C Bindings

```
int PLASMA_cgels(PLASMA_enum trans, int M, int N, int NRHS, PLASMA_Complex32_t *A, int LDA,
    PLASMA_Complex32_t *T, PLASMA_Complex32_t *B, int LDB)
```

6.3 Fortran Bindings

```
PLASMA_CGELS(INTEGER trans, INTEGER M, INTEGER N, INTEGER NRHS, COMPLEX A, INTEGER LDA,
    INTEGER T, COMPLEX B, INTEGER LDB, INTEGER INFO)
```


6.4 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.
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	PLASMA_Complex32_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_cgeqrf; if $M < N$, A is overwritten by details of its LQ factorization as returned by PLASMA_cgelsqf.
LDA	int (IN) The leading dimension of the array A. $LDA \geq \max(1, M)$.
T	PLASMA_Complex32_t* (OUT) On exit, auxiliary factorization data.
B	PLASMA_Complex32_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 \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. $LDB \geq \max(1, M, N)$.

6.5 Return Value:

```

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

```

6.6 Online Browsing

Dive into [PLASMA_cgels](#)

7 PLASMA_cgels_Tile

7.1 Purpose

PLASMA_cgels_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.

7.2 C Bindings

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

7.3 Fortran Bindings

```
PLASMA_CGELS_TILE(INTEGER trans, INTEGER*4 A, INTEGER*4 B, INTEGER*4 T, INTEGER INFO)
```

7.4 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_Complex32_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_cgeqrf; if $M < N$, A is overwritten by details of its LQ factorization as returned by PLASMA_cgelqf.
T	PLASMA_Complex32_t* (OUT) On exit, auxiliary factorization data.
B	PLASMA_Complex32_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 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;

7.5 Return Value:

```
= 0: successful exit
```

7.6 Online Browsing

Dive into [PLASMA_cgels_Tile](#)

8 PLASMA_cgemmm

8.1 Purpose

PLASMA_cgemmm - Performs one of the matrix-matrix operations

$$C = \alpha * \text{op}(A) * \text{op}(B) + \beta * C,$$

where $\text{op}(X)$ is one of

$$\text{op}(X) = X \quad \text{or} \quad \text{op}(X) = X'$$

α and β are scalars, and A , B and C are matrices, with $\text{op}(A)$ an m by k matrix, $\text{op}(B)$ a k by n matrix and C an m by n matrix.

8.2 C Bindings

```
int PLASMA_cgemmm(PLASMA_enum transA, PLASMA_enum transB, int M, int N, int K, ↵
    PLASMA_Complex32_t alpha, PLASMA_Complex32_t *A, int LDA, PLASMA_Complex32_t *B, int LDB ↵
    , PLASMA_Complex32_t beta, PLASMA_Complex32_t *C, int LDC)
```

8.3 Fortran Bindings

```
PLASMA_CGEMM(INTEGER transA, INTEGER transB, INTEGER M, INTEGER N, INTEGER K, COMPLEX alpha ↵
    , COMPLEX A, INTEGER LDA, COMPLEX B, INTEGER LDB, COMPLEX beta, COMPLEX C, INTEGER LDC, ↵
    INTEGER INFO)
```

8.4 Arguments

```
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.
        Currently only PlasmaNoTrans is supported

transB  PLASMA_enum (IN)
        Specifies whether the matrix B is transposed, not transposed or conjugate ↵
        transposed:
        = PlasmaNoTrans:  B is transposed;
        = PlasmaTrans:   B is not transposed;
```

	<code>= PlasmaConjTrans: B is conjugate transposed. Currently only PlasmaNoTrans is supported</code>
M	<code>int (IN)</code> M specifies the number of rows of the matrix <code>op(A)</code> and of the matrix C. $M \geq 0$.
N	<code>int (IN)</code> N specifies the number of columns of the matrix <code>op(B)</code> and of the matrix C. $N \geq 0$. ↩
K	<code>int (IN)</code> K specifies the number of columns of the matrix <code>op(A)</code> and the number of rows of the matrix <code>op(B)</code> . $K \geq 0$.
alpha	<code>PLASMA_Complex32_t (IN)</code> alpha specifies the scalar alpha
A	<code>PLASMA_Complex32_t* (IN)</code> A is a LDA-by-ka matrix, where ka is K when <code>transA = PlasmaNoTrans</code> , and is M otherwise.
LDA	<code>int (IN)</code> The leading dimension of the array A. $LDA \geq \max(1, M)$.
B	<code>PLASMA_Complex32_t* (IN)</code> B is a LDB-by-kb matrix, where kb is N when <code>transB = PlasmaNoTrans</code> , and is K otherwise.
LDB	<code>int (IN)</code> The leading dimension of the array B. $LDB \geq \max(1, N)$.
beta	<code>PLASMA_Complex32_t (IN)</code> beta specifies the scalar beta
C	<code>PLASMA_Complex32_t* (INOUT)</code> C is a LDC-by-N matrix. On exit, the array is overwritten by the M by N matrix $(\alpha * \text{op}(A) * \text{op}(B) + \beta * C)$ ↩
LDC	<code>int (IN)</code> The leading dimension of the array C. $LDC \geq \max(1, M)$.

8.5 Return Value:

`= 0: successful exit`

8.6 Online Browsing

Dive into [PLASMA_cgemm](#)

9 PLASMA_cgeqrf

9.1 Purpose

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

9.2 C Bindings

```
int PLASMA_cgeqrf(int M, int N, PLASMA_Complex32_t *A, int LDA, PLASMA_Complex32_t *T)
```

9.3 Fortran Bindings

```
PLASMA_CGEQRF(INTEGER M, INTEGER N, COMPLEX A, INTEGER LDA, INTEGER T, INTEGER INFO)
```

9.4 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_Complex32_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 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>PLASMA_Complex32_t*</code> (OUT) On exit, auxiliary factorization data, required by PLASMA_cgeqrs to solve the system of equations.

9.5 Return Value:

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

9.6 Online Browsing

Dive into [PLASMA_cgeqrf](#)

10 PLASMA_cgeqrf_Tile

10.1 Purpose

PLASMA_cgeqrf_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.

10.2 C Bindings

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

10.3 Fortran Bindings

```
PLASMA_CGEQRF_TILE(INTEGER*4 A, INTEGER*4 T, INTEGER INFO)
```

10.4 Arguments

A	PLASMA_Complex32_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 >= N); the elements below the diagonal represent the unitary matrix Q as a product of elementary reflectors stored by tiles.
T	PLASMA_Complex32_t* (OUT) On exit, auxiliary factorization data, required by PLASMA_cgeqrs to solve the system of equations.

10.5 Return Value:

```
= 0: successful exit
```

10.6 Online Browsing

Dive into [PLASMA_cgeqrf_Tile](#)

11 PLASMA_cgeqrs

11.1 Purpose

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

11.2 C Bindings

```
int PLASMA_cgeqrs(int M, int N, int NRHS, PLASMA_Complex32_t *A, int LDA,
  PLASMA_Complex32_t *T, PLASMA_Complex32_t *B, int LDB)
```

11.3 Fortran Bindings

```
PLASMA_CGEQRS(INTEGER M, INTEGER N, INTEGER NRHS, COMPLEX A, INTEGER LDA, INTEGER T,
  COMPLEX B, INTEGER LDB, INTEGER INFO)
```

11.4 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_Complex32_t*</code> (INOUT) Details of the QR factorization of the original matrix A as returned by <code>PLASMA_cgeqrf</code> .
LDA	<code>int</code> (IN) The leading dimension of the array A. $LDA \geq M$.
T	<code>PLASMA_Complex32_t*</code> (IN) Auxiliary factorization data, computed by <code>PLASMA_cgeqrf</code> .
B	<code>PLASMA_Complex32_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. $LDB \geq \max(1, N)$.

11.5 Return Value:

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

11.6 Online Browsing

Dive into [PLASMA_cgeqrs](#)

12 PLASMA_cgeqrs_Tile

12.1 Purpose

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

12.2 C Bindings

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

12.3 Fortran Bindings

```
PLASMA_CGEQRS_TILE(INTEGER*4 A, INTEGER*4 B, INTEGER*4 T, INTEGER INFO)
```

12.4 Arguments

A	PLASMA_Complex32_t* (INOUT) Details of the QR factorization of the original matrix A as returned by <code>PLASMA_cgeqrf</code> .
T	PLASMA_Complex32_t* (IN) Auxiliary factorization data, computed by <code>PLASMA_cgeqrf</code> .
B	PLASMA_Complex32_t* (INOUT) On entry, the m-by-nrhs right hand side matrix B. On exit, the n-by-nrhs solution matrix X.

12.5 Return Value:

= 0: successful exit

12.6 Online Browsing

Dive into [PLASMA_cgeqrs_Tile](#)

13 PLASMA_cgesv

13.1 Purpose

PLASMA_cgesv - 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 $A * X = B$.

13.2 C Bindings

```
int PLASMA_cgesv(int N, int NRHS, PLASMA_Complex32_t *A, int LDA, PLASMA_Complex32_t *L,
    int *IPIV, PLASMA_Complex32_t *B, int LDB)
```

13.3 Fortran Bindings

```
PLASMA_CGESV(INTEGER N, INTEGER NRHS, COMPLEX A, INTEGER LDA, INTEGER LH, INTEGER IPIVH,
    COMPLEX B, INTEGER LDB, INTEGER INFO)
```

13.4 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_Complex32_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	<code>PLASMA_Complex32_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) ↔.
B	<code>PLASMA_Complex32_t*</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.
LDB	<code>int</code> (IN) The leading dimension of the array B. $LDB \geq \max(1, N)$.

13.5 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.

```

13.6 Online Browsing

Dive into [PLASMA_cgesv](#)

14 PLASMA_cgesv_Tile

14.1 Purpose

PLASMA_cgesv_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.

14.2 C Bindings

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

14.3 Fortran Bindings

```
PLASMA_CGESV_TILE(INTEGER*4 A, INTEGER*4 L, INTEGER IPIVH, INTEGER*4 B, INTEGER INFO)
```

14.4 Arguments

A	PLASMA_Complex32_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_Complex32_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_Complex32_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.

14.5 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.

```

14.6 Online Browsing

Dive into [PLASMA_cgesv_Tile](#)

15 PLASMA_cgetrf

15.1 Purpose

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

15.2 C Bindings

```

int PLASMA_cgetrf(int M, int N, PLASMA_Complex32_t *A, int LDA, PLASMA_Complex32_t *L, int ↔
    *IPIV)

```

15.3 Fortran Bindings

```

PLASMA_CGETRF(INTEGER M, INTEGER N, COMPLEX A, INTEGER LDA, INTEGER LH, INTEGER IPIVH, ↔
    INTEGER INFO)

```

15.4 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_Complex32_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_Complex32_t*</code> (OUT) On exit, auxiliary factorization data, related to the tile L factor, required by <code>PLASMA_cgetrs</code> to solve the system of equations.
IPIV	<code>int*</code> (OUT) The pivot indices that define the permutations (not equivalent to LAPACK).

15.5 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.

```

15.6 Online Browsing

Dive into [PLASMA_cgetrf](#)

16 PLASMA_cgetrf_Tile

16.1 Purpose

PLASMA_cgetrf_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.

16.2 C Bindings

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

16.3 Fortran Bindings

```
PLASMA_CGETRF_TILE(INTEGER*4 A, INTEGER*4 L, INTEGER IPIVH, INTEGER INFO)
```

16.4 Arguments

A	PLASMA_Complex32_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_Complex32_t* (OUT) On exit, auxiliary factorization data, related to the tile L factor, required by PLASMA_cgetrs to solve the system of equations.
IPIV	int* (OUT) The pivot indices that define the permutations (not equivalent to LAPACK).

16.5 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.

```

16.6 Online Browsing

Dive into [PLASMA_cgetrf_Tile](#)

17 PLASMA_cgetrs

17.1 Purpose

PLASMA_cgetrs - 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_cgetrf.

17.2 C Bindings

```

int PLASMA_cgetrs(PLASMA_enum uplo, int N, int NRHS, PLASMA_Complex32_t *A, int LDA, ↔
    PLASMA_Complex32_t *L, int *IPIV, PLASMA_Complex32_t *B, int LDB)

```

17.3 Fortran Bindings

```

PLASMA_CGETRS(INTEGER uplo, INTEGER N, INTEGER NRHS, COMPLEX A, INTEGER LDA, INTEGER LH, ↔
    INTEGER IPIVH, COMPLEX B, INTEGER LDB, INTEGER INFO)

```

17.4 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	<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>PLASMA_Complex32_t*</code> (IN) The tile factors L and U from the factorization, computed by <code>PLASMA_cgetrf</code> .
LDA	<code>int</code> (IN) The leading dimension of the array A. $LDA \geq \max(1, N)$.
L	<code>PLASMA_Complex32_t*</code> (IN) Auxiliary factorization data, related to the tile L factor, computed by <code>PLASMA_cgetrf</code> .
IPIV	<code>int*</code> (IN) The pivot indices from <code>PLASMA_cgetrf</code> (not equivalent to LAPACK).
B	<code>PLASMA_Complex32_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. $LDB \geq \max(1, N)$.

17.5 Return Value:

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

17.6 Online Browsing

Dive into [PLASMA_cgetrs](#)

18 PLASMA_cgetrs_Tile

18.1 Purpose

PLASMA_cgetrs_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_cgetrf`. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

18.2 C Bindings

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

18.3 Fortran Bindings

```
PLASMA_CGETRS_TILE(INTEGER*4 A, INTEGER*4 L, INTEGER IPIVH, INTEGER*4 B, INTEGER INFO)
```

18.4 Arguments

A	PLASMA_Complex32_t* (IN) The tile factors L and U from the factorization, computed by PLASMA_cgetrf.
L	PLASMA_Complex32_t* (IN) Auxiliary factorization data, related to the tile L factor, computed by \leftrightarrow PLASMA_cgetrf.
IPIV	int* (IN) The pivot indices from PLASMA_cgetrf (not equivalent to LAPACK).
B	PLASMA_Complex32_t* (INOUT) On entry, the N-by-NRHS matrix of right hand side matrix B. On exit, the solution matrix X.

18.5 Return Value:

= 0: successful exit

18.6 Online Browsing

Dive into [PLASMA_cgetrs_Tile](#)

19 PLASMA_Enable

19.1 Purpose

PLASMA_Enable - Enable PLASMA feature.

19.2 C Bindings

```
int PLASMA_Enable(PLASMA_enum lever)
```

19.3 Fortran Bindings

```
PLASMA_ENABLE(INTEGER lever, INTEGER INFO)
```

19.4 Arguments

lever	PLASMA_enum (IN) Feature to be enabled: = PLASMA_WARNINGS: printing of warning messages, = PLASMA_ERRORS: printing of error messages, = PLASMA_AUTOTUNING: autotuning for tile size and inner block size.
-------	---

19.5 Return Value:

```
= PLASMA_SUCCESS: successful exit
```

19.6 Online Browsing

Dive into [PLASMA_Enable](#)

20 PLASMA_Disable

20.1 Purpose

PLASMA_Disable - Disable PLASMA feature.

20.2 C Bindings

```
int PLASMA_Disable(PLASMA_enum lever)
```

20.3 Fortran Bindings

```
PLASMA_DISABLE(INTEGER lever, INTEGER INFO)
```

20.4 Arguments

lever	PLASMA_enum (IN) Feature to be disabled: = PLASMA_WARNINGS: printing of warning messages, = PLASMA_ERRORS: printing of error messages, = PLASMA_AUTOTUNING: autotuning for tile size and inner block size.
-------	--

20.5 Return Value:

```
= PLASMA_SUCCESS: successful exit
```

20.6 Online Browsing

Dive into [PLASMA_Disable](#)

21 PLASMA_Set

21.1 Purpose

PLASMA_Set - Set PLASMA parameter

21.2 C Bindings

```
int PLASMA_Set (PLASMA_enum param, int value)
```

21.3 Fortran Bindings

```
PLASMA_SET (INTEGER param, INTEGER value, INTEGER INFO)
```

21.4 Arguments

```
param    PLASMA_enum (IN)
          PLASMA parameter:
          = PLASMA_TILE_SIZE:      size matrix tile,
          = PLASMA_INNER_BLOCK_SIZE: size of tile inner block.

value    int (IN)
          Value of the parameter.
```

21.5 Return Value:

```
= PLASMA_SUCCESS: successful exit
```

21.6 Online Browsing

Dive into [PLASMA_Set](#)

22 PLASMA_Get

22.1 Purpose

PLASMA_Get - Get value of PLASMA parameter

22.2 C Bindings

```
int PLASMA_Get (PLASMA_enum param, int *value)
```

22.3 Fortran Bindings

```
PLASMA_GET (INTEGER param, INTEGER value, INTEGER INFO)
```

22.4 Arguments

```
param    PLASMA_enum (IN)
          PLASMA parameter:
          = PLASMA_TILE_SIZE:      size matrix tile,
          = PLASMA_INNER_BLOCK_SIZE: size of tile inner block.

value    int* (OUT)
          Value of the parameter.
```


22.5 Return Value:

```
= PLASMA_SUCCESS: successful exit
```

22.6 Online Browsing

Dive into [PLASMA_Get](#)

23 PLASMA_Init

23.1 Purpose

PLASMA_Init - Initialize PLASMA.

23.2 C Bindings

```
int PLASMA_Init(int cores)
```

23.3 Fortran Bindings

```
PLASMA_INIT(INTEGER CORES, INTEGER INFO)
```

23.4 Arguments

cores	<code>int</code> (IN)
	Number of cores to use (threads to launch)

23.5 Return Value:

```
= PLASMA_SUCCESS: successful exit
```

23.6 Online Browsing

Dive into [PLASMA_Init](#)

24 PLASMA_Finalize

24.1 Purpose

PLASMA_Finalize - Finalize PLASMA.

24.2 C Bindings

```
int PLASMA_Finalize()
```

24.3 Fortran Bindings

```
PLASMA_FINALIZE (INTEGER INFO)
```

24.4 Online Browsing

Dive into [PLASMA_Finalize](#)

25 PLASMA_cposv

25.1 Purpose

PLASMA_cposv - 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 $A * X = B$.

25.2 C Bindings

```
int PLASMA_cposv(PLASMA_enum uplo, int N, int NRHS, PLASMA_Complex32_t *A, int LDA, ↵
    PLASMA_Complex32_t *B, int LDB)
```

25.3 Fortran Bindings

```
PLASMA_CPOSV(INTEGER uplo, INTEGER N, INTEGER NRHS, COMPLEX A, INTEGER LDA, COMPLEX B, ↵
    INTEGER LDB, INTEGER INFO)
```

25.4 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_Complex32_t* (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 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_Complex32_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. $LDB \geq \max(1, N)$.

25.5 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

25.6 Online Browsing

Dive into [PLASMA_cposv](#)

26 PLASMA_cposv_Tile

26.1 Purpose

PLASMA_cposv_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

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

26.2 C Bindings

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

26.3 Fortran Bindings

```
PLASMA_CPOSV_TILE(INTEGER uplo, INTEGER*4 A, INTEGER*4 B, INTEGER INFO)
```

26.4 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_Complex32_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_Complex32_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.

26.5 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
    .

```

26.6 Online Browsing

Dive into [PLASMA_cposv_Tile](#)

27 PLASMA_cpotrf

27.1 Purpose

PLASMA_cpotrf - 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.

27.2 C Bindings

```
int PLASMA_cpotrf(PLASMA_enum uplo, int N, PLASMA_Complex32_t *A, int LDA)
```

27.3 Fortran Bindings

```
PLASMA_CPOTRF(INTEGER uplo, INTEGER N, COMPLEX A, INTEGER LDA, INTEGER INFO)
```

27.4 Arguments

uplo	PLASMA_enum (IN) = PlasmaUpper: Upper triangle of A is stored; = PlasmaLower: Lower triangle of A is stored.
N	<code>int</code> (IN) The order of the matrix A. $N \geq 0$.
A	PLASMA_Complex32_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 <code>return</code> value = 0, the factor U or L from the Cholesky factorization $A = U^*H^*U$ or $A = L^*L^*H$.
LDA	<code>int</code> (IN) The leading dimension of the array A. $LDA \geq \max(1, N)$.

27.5 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
```

27.6 Online Browsing

Dive into [PLASMA_cpotrf](#)

28 PLASMA_cpotrf_Tile

28.1 Purpose

PLASMA_cpotrf_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.

28.2 C Bindings

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

28.3 Fortran Bindings

```
PLASMA_CPOTRF_TILE(INTEGER uplo, INTEGER*4 A, INTEGER INFO)
```

28.4 Arguments

uplo	PLASMA_enum (IN) = PlasmaUpper: Upper triangle of A is stored; = PlasmaLower: Lower triangle of A is stored.
A	PLASMA_Complex32_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 $A = L^*L^*H$.

28.5 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
    .
```

28.6 Online Browsing

Dive into [PLASMA_cpotrf_Tile](#)

29 PLASMA_cpotrs

29.1 Purpose

PLASMA_cpotrs - 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_cpotrf.

29.2 C Bindings

```
int PLASMA_cpotrs(PLASMA_enum uplo, int N, int NRHS, PLASMA_Complex32_t *A, int LDA,
    PLASMA_Complex32_t *B, int LDB)
```

29.3 Fortran Bindings

```
PLASMA_CPOTRS(INTEGER uplo, INTEGER N, INTEGER NRHS, COMPLEX A, INTEGER LDA, COMPLEX B, ←
              INTEGER LDB, INTEGER INFO)
```

29.4 Arguments

uplo	PLASMA_enum (IN) = PlasmaUpper: Upper triangle of A is stored; = PlasmaLower: Lower triangle of A is stored.
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	PLASMA_Complex32_t* (IN) The triangular factor U or L from the Cholesky factorization $A = U^*H^*U$ or $A = L^*L$ ← **H , computed by PLASMA_cpotrf.
LDA	<code>int</code> (IN) The leading dimension of the array A. $LDA \geq \max(1, N)$.
B	PLASMA_Complex32_t* (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>int</code> (IN) The leading dimension of the array B. $LDB \geq \max(1, N)$.

29.5 Return Value:

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

29.6 Online Browsing

Dive into [PLASMA_cpotrs](#)

30 PLASMA_cpotrs_Tile

30.1 Purpose

PLASMA_cpotrs_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_cpotrf. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

30.2 C Bindings

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

30.3 Fortran Bindings

```
PLASMA_CPOTRS_TILE(INTEGER uplo, INTEGER*4 A, INTEGER*4 B, INTEGER INFO)
```

30.4 Arguments

uplo	PLASMA_enum (IN) = PlasmaUpper: Upper triangle of A is stored; = PlasmaLower: Lower triangle of A is stored.
A	PLASMA_Complex32_t* (IN) The triangular factor U or L from the Cholesky factorization $A = U^*H^*U$ or $A = L^*L$ ← **H, computed by PLASMA_cpotrf.
B	PLASMA_Complex32_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.

30.5 Return Value:

```
= 0: successful exit
```

30.6 Online Browsing

Dive into [PLASMA_cpotrs_Tile](#)

31 PLASMA_ctrsm

31.1 Purpose

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

31.2 C Bindings

```
int PLASMA_ctrsm(PLASMA_enum side, PLASMA_enum uplo, PLASMA_enum transA, PLASMA_enum diag, ←  
    int N, int NRHS, PLASMA_Complex32_t *A, int LDA, PLASMA_Complex32_t *B, int LDB)
```

31.3 Fortran Bindings

```
PLASMA_CTRSM(INTEGER side, INTEGER uplo, INTEGER transA, INTEGER diag, INTEGER N, INTEGER ←  
    NRHS, COMPLEX A, INTEGER LDA, COMPLEX B, INTEGER LDB, INTEGER INFO)
```


31.4 Arguments

side	PLASMA_enum (IN) Specifies whether A appears on the left or on the right of X: = PlasmaLeft: $A \cdot X = B$ = PlasmaRight: $X \cdot A = B$
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_Complex32_t* (IN) The triangular matrix A. If uplo = PlasmaUpper, the leading N-by-N upper triangular \leftrightarrow 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_Complex32_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_Complex32_t* (IN) The leading dimension of the array B. $LDB \geq \max(1, N)$.

31.5 Return Value:

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

31.6 Online Browsing

Dive into [PLASMA_ctrsm](#)

32 PLASMA_ctrsm_Tile

32.1 Purpose

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

32.2 C Bindings

```
int PLASMA_ctrsm_Tile(PLASMA_enum side, PLASMA_enum uplo, PLASMA_enum transA, PLASMA_enum ←
diag, PLASMA_desc *A, PLASMA_desc *B)
```

32.3 Fortran Bindings

```
PLASMA_CTRSM_TILE(INTEGER side, INTEGER uplo, INTEGER transA, INTEGER diag, INTEGER*4 A, ←
INTEGER*4 B, INTEGER INFO)
```

32.4 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 is unit.
A	PLASMA_Complex32_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_Complex32_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.

32.5 Return Value:

= 0: successful exit

32.6 Online Browsing

Dive into [PLASMA_ctrsm_Tile](#)

33 PLASMA_ctrsmpl

33.1 Purpose

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

33.2 C Bindings

```
int PLASMA_ctrsmpl(int N, int NRHS, PLASMA_Complex32_t *A, int LDA, PLASMA_Complex32_t *L,
    int *IPIV, PLASMA_Complex32_t *B, int LDB)
```

33.3 Fortran Bindings

```
PLASMA_CTRSMPL(INTEGER N, INTEGER NRHS, COMPLEX A, INTEGER LDA, INTEGER LH, INTEGER IPIVH,
    COMPLEX B, INTEGER LDB, INTEGER INFO)
```

33.4 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>PLASMA_Complex32_t*</code> (IN) The tile factor L from the factorization, computed by <code>PLASMA_cgetrf</code> .
LDA	<code>int</code> (IN) The leading dimension of the array A. $LDA \geq \max(1, N)$.
L	<code>PLASMA_Complex32_t*</code> (IN) Auxiliary factorization data, related to the tile L factor, computed by <code>PLASMA_cgetrf</code> .
IPIV	<code>int*</code> (IN) The pivot indices from <code>PLASMA_cgetrf</code> (not equivalent to LAPACK).
B	<code>PLASMA_Complex32_t*</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.
LDB	<code>PLASMA_Complex32_t*</code> (IN) The leading dimension of the array B. $LDB \geq \max(1, N)$.

33.5 Return Value:

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

33.6 Online Browsing

Dive into [PLASMA_ctrsmpl](#)

34 PLASMA_ctrsmpl_Tile

34.1 Purpose

PLASMA_ctrsmpl_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.

34.2 C Bindings

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

34.3 Fortran Bindings

```
PLASMA_CTRSMPL_TILE(INTEGER*4 A, INTEGER*4 L, INTEGER IPIVH, INTEGER*4 B, INTEGER INFO)
```

34.4 Arguments

A	PLASMA_Complex32_t* (IN) The tile factor L from the factorization, computed by PLASMA_cgetrf.
L	PLASMA_Complex32_t* (IN) Auxiliary factorization data, related to the tile L factor, computed by \leftrightarrow PLASMA_cgetrf.
IPIV	int* (IN) The pivot indices from PLASMA_cgetrf (not equivalent to LAPACK).
B	PLASMA_Complex32_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.

34.5 Return Value:

```
= 0: successful exit
```

34.6 Online Browsing

Dive into [PLASMA_ctrsmpl_Tile](#)

35 PLASMA_cunglq

35.1 Purpose

PLASMA_cunglq - 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_cgelsqf.

35.2 C Bindings

```
int PLASMA_cunglq(int M, int N, int K, PLASMA_Complex32_t *A, int LDA, PLASMA_Complex32_t * ←
T, PLASMA_Complex32_t *B, int LDB)
```

35.3 Fortran Bindings

```
PLASMA_CUNGLQ(INTEGER M, INTEGER N, INTEGER K, COMPLEX A, INTEGER LDA, INTEGER T, COMPLEX B ←
, INTEGER LDB, INTEGER INFO)
```

35.4 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 rows of elementary tile reflectors whose product defines the matrix ← Q. $M \geq K \geq 0$.
A	PLASMA_Complex32_t* (IN) Details of the LQ factorization of the original matrix A as returned by ← PLASMA_cgelsqf.
LDA	<code>int</code> (IN) The leading dimension of the array A. $LDA \geq \max(1, M)$.
T	PLASMA_Complex32_t* (IN) Auxiliary factorization data, computed by PLASMA_cgelsqf.
B	PLASMA_Complex32_t* (OUT) On exit, the M-by-N matrix Q.
LDB	<code>int</code> (IN) The leading dimension of the array B. $LDB \geq \max(1, M)$.

35.5 Return Value:

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

35.6 Online Browsing

Dive into [PLASMA_cunglq](#)

36 PLASMA_cunglq_Tile

36.1 Purpose

PLASMA_cunglq_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_cgelqf_Tile. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

36.2 C Bindings

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

36.3 Fortran Bindings

36.4 Arguments

A	PLASMA_Complex32_t* (IN) Details of the LQ factorization of the original matrix A as returned by <code>PLASMA_cgelqf</code> .
T	PLASMA_Complex32_t* (IN) Auxiliary factorization data, computed by PLASMA_cgelqf.
B	PLASMA_Complex32_t* (OUT) On exit, the M-by-N matrix Q.

36.5 Return Value:

```
= 0: successful exit
```

36.6 Online Browsing

Dive into [PLASMA_cunglq_Tile](#)

37 PLASMA_cungqr

37.1 Purpose

PLASMA_cungqr - 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_cgeqrf.

37.2 C Bindings

```
int PLASMA_cungqr(int M, int N, int K, PLASMA_Complex32_t *A, int LDA, PLASMA_Complex32_t * ←
    T, PLASMA_Complex32_t *B, int LDB)
```

37.3 Fortran Bindings

```
PLASMA_CUNGQR(INTEGER M, INTEGER N, INTEGER K, COMPLEX A, INTEGER LDA, INTEGER T, COMPLEX B ←
    , INTEGER LDB, INTEGER INFO)
```

37.4 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	<code>PLASMA_Complex32_t*</code> (IN) Details of the QR factorization of the original matrix A as returned by ← <code>PLASMA_cgeqrf</code> .
LDA	<code>int</code> (IN) The leading dimension of the array A. $LDA \geq \max(1, M)$.
T	<code>PLASMA_Complex32_t*</code> (IN) Auxiliary factorization data, computed by <code>PLASMA_cgeqrf</code> .
B	<code>PLASMA_Complex32_t*</code> (OUT) On exit, the M-by-N matrix Q.
LDB	<code>int</code> (IN) The leading dimension of the array B. $LDB \geq \max(1, M)$.

37.5 Return Value:

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

37.6 Online Browsing

Dive into [PLASMA_cungqr](#)

38 PLASMA_cungqr_Tile

38.1 Purpose

PLASMA_cungqr_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_cgeqrf_Tile. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

38.2 C Bindings

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

38.3 Fortran Bindings

```
PLASMA_CUNGQR_TILE(INTEGER*4 A, INTEGER*4 T, INTEGER*4 B, INTEGER INFO)
```

38.4 Arguments

A	PLASMA_Complex32_t* (IN) Details of the QR factorization of the original matrix A as returned by <code>PLASMA_cgeqrf</code> .
T	PLASMA_Complex32_t* (IN) Auxiliary factorization data, computed by <code>PLASMA_cgeqrf</code> .
B	PLASMA_Complex32_t* (OUT) On exit, the M-by-N matrix Q.

38.5 Return Value:

```
= 0: successful exit
```

38.6 Online Browsing

Dive into [PLASMA_cungqr_Tile](#)

39 PLASMA_cunmlq

39.1 Purpose

PLASMA_cunmlq - 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_cgelqf`. Q is of order M.

39.2 C Bindings

```
int PLASMA_cunmlq(PLASMA_enum side, PLASMA_enum trans, int M, int N, int K,   
  PLASMA_Complex32_t *A, int LDA, PLASMA_Complex32_t *T, PLASMA_Complex32_t *B, int LDB)
```


39.3 Fortran Bindings

```
PLASMA_CUNMLQ(INTEGER side, INTEGER trans, INTEGER M, INTEGER N, INTEGER K, COMPLEX A, ↵
               INTEGER LDA, INTEGER T, COMPLEX B, INTEGER LDB, INTEGER INFO)
```

39.4 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 ↵ Q. $M \geq K \geq 0$.
A	PLASMA_Complex32_t* (IN) Details of the LQ factorization of the original matrix A as returned by ↵ PLASMA_cgelsqf.
LDA	<code>int</code> (IN) The leading dimension of the array A. $LDA \geq \max(1, K)$.
T	PLASMA_Complex32_t* (IN) Auxiliary factorization data, computed by PLASMA_cgelsqf.
B	PLASMA_Complex32_t* (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. $LDB \geq \max(1, M)$.

39.5 Return Value:

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

39.6 Online Browsing

Dive into [PLASMA_cunmlq](#)

40 PLASMA_cunmlq_Tile

40.1 Purpose

PLASMA_cunmlq_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_cgelsqf_Tile** Q is of order M. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

40.2 C Bindings

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

40.3 Fortran Bindings

```
PLASMA_CUNMLQ_TILE(INTEGER side, INTEGER trans, INTEGER*4 A, INTEGER*4 T, INTEGER*4 B, ←
    INTEGER INFO)
```

40.4 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_Complex32_t* (IN) Details of the LQ factorization of the original matrix A as returned by ← PLASMA_cgelsqf.
T	PLASMA_Complex32_t* (IN) Auxiliary factorization data, computed by PLASMA_cgelsqf.
B	PLASMA_Complex32_t* (INOUT) On entry, the M-by-N matrix B. On exit, B is overwritten by Q^*B or Q^*H^*B .

40.5 Return Value:

= 0: successful exit

40.6 Online Browsing

Dive into [PLASMA_cunmlq_Tile](#)

41 PLASMA_cunmqr

41.1 Purpose

PLASMA_cunmqr - 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_cgeqrf. Q is of order M.

41.2 C Bindings

```
int PLASMA_cunmqr(PLASMA_enum side, PLASMA_enum trans, int M, int N, int K, ↵
    PLASMA_Complex32_t *A, int LDA, PLASMA_Complex32_t *T, PLASMA_Complex32_t *B, int LDB)
```

41.3 Fortran Bindings

```
PLASMA_CUNMQR(INTEGER side, INTEGER trans, INTEGER M, INTEGER N, INTEGER K, COMPLEX A, ↵
    INTEGER LDA, INTEGER T, COMPLEX B, INTEGER LDB, INTEGER INFO)
```

41.4 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 ↵ matrix Q. $M \geq K \geq 0$.
A	PLASMA_Complex32_t* (IN) Details of the QR factorization of the original matrix A as returned by ↵ PLASMA_cgeqrf.
LDA	int (IN) The leading dimension of the array A. $LDA \geq \max(1, M)$;
T	PLASMA_Complex32_t* (IN) Auxiliary factorization data, computed by PLASMA_cgeqrf.
B	PLASMA_Complex32_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. LDC >= max(1,M).
```

41.5 Return Value:

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

41.6 Online Browsing

Dive into [PLASMA_cunmqr](#)

42 PLASMA_cunmqr_Tile

42.1 Purpose

PLASMA_cunmqr_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_cgeqrf_Tile** Q is of order M. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

42.2 C Bindings

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

42.3 Fortran Bindings

```
PLASMA_CUNMQR_TILE(INTEGER side, INTEGER trans, INTEGER*4 A, INTEGER*4 T, INTEGER*4 B, ←
    INTEGER INFO)
```

42.4 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_Complex32_t* (IN) Details of the QR factorization of the original matrix A as returned by ← PLASMA_cgeqrf.
T	PLASMA_Complex32_t* (IN)

Auxiliary factorization data, computed by PLASMA_cgeqrf.

B PLASMA_Complex32_t* (INOUT)
On entry, the M-by-N matrix B.
On exit, B is overwritten by $Q*B$ or $Q**H*B$.

42.5 Return Value:

= 0: successful exit

42.6 Online Browsing

Dive into [PLASMA_cunmqr_Tile](#)

43 PLASMA_Desc_Create

43.1 Purpose

PLASMA_Desc_Create - Create matrix descriptor.

43.2 C Bindings

```
int PLASMA_Desc_Create(PLASMA_desc **desc, void *mat, PLASMA_enum dtyp, int mb, int nb, int ↵
    bsiz, int lm, int ln, int i, int j, int m, int n)
```

43.3 Fortran Bindings

```
PLASMA_DESC_CREATE(INTEGER*4 desc, INTEGER mat, INTEGER dtyp, INTEGER mb, INTEGER nb, ↵
    INTEGER bsiz, INTEGER lm, INTEGER ln, INTEGER i, INTEGER j, INTEGER m, INTEGER n, ↵
    INTEGER INFO)
```

43.4 Arguments

desc	PLASMA_desc** (OUT) On exit, descriptor of the matrix.
mat	void* (IN) Memory location of the matrix.
dtyp	PLASMA_enum (IN) Data type of the matrix: = PlasmaRealFloat: single precision real (S), = PlasmaRealDouble: double precision real (D), = PlasmaComplexFloat: single precision complex (C), = PlasmaComplexDouble: double precision complex (Z).
mb	int (IN) Number of rows in a tile.
nb	int (IN)

	Number of columns in a tile.
bsiz	<code>int</code> (IN) Nize in elements including padding.
lm	<code>int</code> (IN) Number of rows of the entire matrix.
ln	<code>int</code> (IN) Number of columns of the entire matrix.
i	<code>int</code> (IN) Row index to the beginning of the submatrix.
j	<code>int</code> (IN) Column indes to the beginning of the submatrix.
m	<code>int</code> (IN) Number of rows of the submatrix.
n	<code>int</code> (IN) Number of columns of the submatrix.

43.5 Return Value:

= PLASMA_SUCCESS: successful exit

43.6 Online Browsing

Dive into [PLASMA_Desc_Create](#)

44 PLASMA_Desc_Destroy

44.1 Purpose

PLASMA_Desc_Destroy - Destroys matrix descriptor.

44.2 C Bindings

```
int PLASMA_Desc_Destroy(PLASMA_desc **desc)
```

44.3 Fortran Bindings

```
PLASMA_DESC_DESTROY(INTEGER*4 desc, INTEGER INFO)
```

44.4 Arguments

desc	PLASMA_desc** (IN) Matrix descriptor.
------	--

44.5 Return Value:

```
= PLASMA_SUCCESS: successful exit
```

44.6 Online Browsing

Dive into [PLASMA_Desc_Destroy](#)

45 PLASMA_dgelqf

45.1 Purpose

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

45.2 C Bindings

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

45.3 Fortran Bindings

```
PLASMA_DGELQF(INTEGER M, INTEGER N, DOUBLE PRECISION A, INTEGER LDA, INTEGER T, INTEGER ↵
INFO)
```

45.4 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.

45.5 Return Value:

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

45.6 Online Browsing

Dive into [PLASMA_dgelqf](#)

46 PLASMA_dgelqf_Tile

46.1 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.

46.2 C Bindings

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

46.3 Fortran Bindings

```
PLASMA_DGELQF_TILE(INTEGER*4 A, INTEGER*4 T, INTEGER INFO)
```

46.4 Arguments

A	double* (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(\leftrightarrow M,N) lower trapezoidal matrix L (L is lower triangular if $M \leq N$); the elements above \leftrightarrow the diagonal represent the unitary matrix Q as a product of elementary reflectors, \leftrightarrow stored by tiles.
T	double* (OUT) On exit, auxiliary factorization data, required by PLASMA_dgelqs to solve the \leftrightarrow system of equations.

46.5 Return Value:

```
= 0: successful exit
```

46.6 Online Browsing

Dive into [PLASMA_dgelqf_Tile](#)

47 PLASMA_dgelqs

47.1 Purpose

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

47.2 C Bindings

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

47.3 Fortran Bindings

```
PLASMA_DGELQS(INTEGER M, INTEGER N, INTEGER NRHS, DOUBLE PRECISION A, INTEGER LDA, INTEGER ↵
T, DOUBLE PRECISION B, INTEGER LDB, INTEGER INFO)
```

47.4 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 ↵ PLASMA_dgelqf.
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_dgelqf.
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. $LDB \geq N$.

47.5 Return Value:

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

47.6 Online Browsing

Dive into [PLASMA_dgelqs](#)

48 PLASMA_dgelqs_Tile

48.1 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.

48.2 C Bindings

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

48.3 Fortran Bindings

```
PLASMA_DGELQS_TILE(INTEGER*4 A, INTEGER*4 B, INTEGER*4 T, INTEGER INFO)
```

48.4 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.

48.5 Return Value:

```
= 0: successful exit
```

48.6 Online Browsing

Dive into [PLASMA_dgelqs_Tile](#)

49 PLASMA_dgels

49.1 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 .

49.2 C Bindings

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

49.3 Fortran Bindings

```
PLASMA_DGELS(INTEGER trans, INTEGER M, INTEGER N, INTEGER NRHS, DOUBLE PRECISION A, INTEGER ←
LDA, INTEGER T, DOUBLE PRECISION B, INTEGER LDB, INTEGER INFO)
```

49.4 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 ← 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 ← 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).

```

49.5 Return Value:

```

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

```

49.6 Online Browsing

Dive into [PLASMA_dgels](#)

50 PLASMA_dgels_Tile

50.1 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.

50.2 C Bindings

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

50.3 Fortran Bindings

```
PLASMA_DGELS_TILE(INTEGER trans, INTEGER*4 A, INTEGER*4 B, INTEGER*4 T, INTEGER INFO)
```

50.4 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 >= 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 >= 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;

```

50.5 Return Value:

```
= 0: successful exit
```

50.6 Online Browsing

Dive into [PLASMA_dgels_Tile](#)

51 PLASMA_dgemm

51.1 Purpose

PLASMA_dgemm - Performs one of the matrix-matrix operations

$$C = \alpha * \text{op}(A) * \text{op}(B) + \beta * C,$$

where $\text{op}(X)$ is one of

$$\text{op}(X) = X \quad \text{or} \quad \text{op}(X) = X'$$

α and β are scalars, and A, B and C are matrices, with $\text{op}(A)$ an m by k matrix, $\text{op}(B)$ a k by n matrix and C an m by n matrix.

51.2 C Bindings

```

int PLASMA_dgemm(PLASMA_enum transA, PLASMA_enum transB, int M, int N, int K, double alpha, ↵
    double *A, int LDA, double *B, int LDB, double beta, double *C, int LDC)

```

51.3 Fortran Bindings

```
PLASMA_DGEMM(INTEGER transA, INTEGER transB, INTEGER M, INTEGER N, INTEGER K, DOUBLE ←
  PRECISION alpha, DOUBLE PRECISION A, INTEGER LDA, DOUBLE PRECISION B, INTEGER LDB, ←
  DOUBLE PRECISION beta, DOUBLE PRECISION C, INTEGER LDC, INTEGER INFO)
```

51.4 Arguments

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. Currently only PlasmaNoTrans is supported
transB	PLASMA_enum (IN) Specifies whether the matrix B is transposed, not transposed or conjugate ← transposed: = PlasmaNoTrans: B is transposed; = PlasmaTrans: B is not transposed; = PlasmaTrans: B is conjugate transposed. Currently only PlasmaNoTrans is supported
M	int (IN) M specifies the number of rows of the matrix op(A) and of the matrix C. M >= 0.
N	int (IN) N specifies the number of columns of the matrix op(B) and of the matrix C. N >= ← 0.
K	int (IN) K specifies the number of columns of the matrix op(A) and the number of rows of the matrix op(B). K >= 0.
alpha	double (IN) alpha specifies the scalar alpha
A	double* (IN) A is a LDA-by-ka matrix, where ka is K when transA = PlasmaNoTrans, and is M otherwise.
LDA	int (IN) The leading dimension of the array A. LDA >= max(1,M).
B	double* (IN) B is a LDB-by-kb matrix, where kb is N when transB = PlasmaNoTrans, and is K otherwise.
LDB	int (IN) The leading dimension of the array B. LDB >= max(1,N).
beta	double (IN) beta specifies the scalar beta
C	double* (INOUT) C is a LDC-by-N matrix. On exit, the array is overwritten by the M by N matrix (alpha*op(A)*op(B) + ← beta*C)

LDC `int` (IN)
 The leading dimension of the array C. `LDC >= max(1,M)`.

51.5 Return Value:

`= 0`: successful exit

51.6 Online Browsing

Dive into [PLASMA_dgemm](#)

52 PLASMA_dgeqrf

52.1 Purpose

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

52.2 C Bindings

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

52.3 Fortran Bindings

```
PLASMA_DGEQRF(INTEGER M, INTEGER N, DOUBLE PRECISION A, INTEGER LDA, INTEGER T, INTEGER ←  

INFO)
```

52.4 Arguments

M	<code>int</code> (IN) The number of rows of the matrix A. <code>M >= 0</code> .
N	<code>int</code> (IN) The number of columns of the matrix A. <code>N >= 0</code> .
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 <code>if M >= N</code>); 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. <code>LDA >= max(1,M)</code> .
T	<code>double*</code> (OUT) On exit, auxiliary factorization data, required by PLASMA_dgeqrs to solve the ← system of equations.

52.5 Return Value:

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

52.6 Online Browsing

Dive into [PLASMA_dgeqrf](#)

53 PLASMA_dgeqrf_Tile

53.1 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.

53.2 C Bindings

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

53.3 Fortran Bindings

```
PLASMA_DGEQRF_TILE(INTEGER*4 A, INTEGER*4 T, INTEGER INFO)
```

53.4 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.	↔

53.5 Return Value:

```
= 0: successful exit
```

53.6 Online Browsing

Dive into [PLASMA_dgeqrf_Tile](#)

54 PLASMA_dgeqrs

54.1 Purpose

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

54.2 C Bindings

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

54.3 Fortran Bindings

```
PLASMA_DGEQRS(INTEGER M, INTEGER N, INTEGER NRHS, DOUBLE PRECISION A, INTEGER LDA, INTEGER ↵
T, DOUBLE PRECISION B, INTEGER LDB, INTEGER INFO)
```

54.4 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 ↵ PLASMA_dgeqrf.
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. $LDB \geq \max(1,N)$.

54.5 Return Value:

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

54.6 Online Browsing

Dive into [PLASMA_dgeqrs](#)

55 PLASMA_dgeqrs_Tile

55.1 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.

55.2 C Bindings

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

55.3 Fortran Bindings

```
PLASMA_DGEQRS_TILE(INTEGER*4 A, INTEGER*4 B, INTEGER*4 T, INTEGER INFO)
```

55.4 Arguments

A	<code>double*</code> (INOUT) Details of the QR factorization of the original matrix A as returned by <code>PLASMA_dgeqrf</code> .
T	<code>double*</code> (IN) Auxiliary factorization data, computed by <code>PLASMA_dgeqrf</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.

55.5 Return Value:

= 0: successful exit

55.6 Online Browsing

Dive into [PLASMA_dgeqrs_Tile](#)

56 PLASMA_dgesv

56.1 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 $A * X = B$.

56.2 C Bindings

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

56.3 Fortran Bindings

```
PLASMA_DGESV(INTEGER N, INTEGER NRHS, DOUBLE PRECISION A, INTEGER LDA, INTEGER LH, INTEGER IPIVH, DOUBLE PRECISION B, INTEGER LDB, INTEGER INFO) ↵
```

56.4 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 ↵).)
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) ↵. .
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. $LDB \geq \max(1, N)$.

56.5 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.
```

56.6 Online Browsing

Dive into [PLASMA_dgesv](#)

57 PLASMA_dgesv_Tile

57.1 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.

57.2 C Bindings

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

57.3 Fortran Bindings

```
PLASMA_DGESV_TILE(INTEGER*4 A, INTEGER*4 L, INTEGER IPIVH, INTEGER*4 B, INTEGER INFO)
```

57.4 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, <code>if return</code> value = 0, the N-by-NRHS solution matrix X.

57.5 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.
```

57.6 Online Browsing

Dive into [PLASMA_dgesv_Tile](#)

58 PLASMA_dgetrf

58.1 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.

58.2 C Bindings

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

58.3 Fortran Bindings

```
PLASMA_DGETRF(INTEGER M, INTEGER N, DOUBLE PRECISION A, INTEGER LDA, INTEGER LH, INTEGER ←  
IPIVH, INTEGER INFO)
```

58.4 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).

58.5 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.
```

58.6 Online Browsing

Dive into [PLASMA_dgetrf](#)

59 PLASMA_dgetrf_Tile

59.1 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.

59.2 C Bindings

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

59.3 Fortran Bindings

```
PLASMA_DGETRF_TILE(INTEGER*4 A, INTEGER*4 L, INTEGER IPIVH, INTEGER INFO)
```

59.4 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).

59.5 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.
```

59.6 Online Browsing

Dive into [PLASMA_dgetrf_Tile](#)

60 PLASMA_dgetrs

60.1 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.

60.2 C Bindings

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

60.3 Fortran Bindings

```
PLASMA_DGETTRS(INTEGER uplo, INTEGER N, INTEGER NRHS, DOUBLE PRECISION A, INTEGER LDA, ←
    INTEGER LH, INTEGER IPIVH, DOUBLE PRECISION B, INTEGER LDB, INTEGER INFO)
```

60.4 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 ← 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. $LDB \geq \max(1, N)$.

60.5 Return Value:

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

60.6 Online Browsing

Dive into [PLASMA_dgettrs](#)

61 PLASMA_dgetrs_Tile

61.1 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.

61.2 C Bindings

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

61.3 Fortran Bindings

```
PLASMA_DGETRS_TILE(INTEGER*4 A, INTEGER*4 L, INTEGER IPIVH, INTEGER*4 B, INTEGER INFO)
```

61.4 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 <code>PLASMA_dgetrf</code> .
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.

61.5 Return Value:

```
= 0: successful exit
```

61.6 Online Browsing

Dive into [PLASMA_dgetrs_Tile](#)

62 PLASMA_dposv

62.1 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 $A * X = B$.

62.2 C Bindings

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

62.3 Fortran Bindings

```
PLASMA_DPOSV(INTEGER uplo, INTEGER N, INTEGER NRHS, DOUBLE PRECISION A, INTEGER LDA, DOUBLE ←  
PRECISION B, INTEGER LDB, INTEGER INFO)
```

62.4 Arguments

<code>uplo</code>	<p><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.</p>
<code>N</code>	<p><code>int</code> (IN) The number of linear equations, i.e., the order of the matrix A. $N \geq 0$.</p>
<code>NRHS</code>	<p><code>int</code> (IN) The number of right hand sides, i.e., the number of columns of the matrix B. <code>NRHS</code> ← ≥ 0.</p>
<code>A</code>	<p><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. 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, <code>if return</code> value = 0, the factor U or L from the Cholesky factorization $A = U^{**T}U$ or $A = L^{**T}L$.</p>
<code>LDA</code>	<p><code>int</code> (IN) The leading dimension of the array A. $LDA \geq \max(1, N)$.</p>
<code>B</code>	<p><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.</p>
<code>LDB</code>	<p><code>int</code> (IN) The leading dimension of the array B. $LDB \geq \max(1, N)$.</p>

62.5 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 ↵
      .

```

62.6 Online Browsing

Dive into [PLASMA_dposv](#)

63 PLASMA_dposv_Tile

63.1 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.

63.2 C Bindings

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

63.3 Fortran Bindings

```
PLASMA_DPOSV_TILE(INTEGER uplo, INTEGER*4 A, INTEGER*4 B, INTEGER INFO)
```

63.4 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.

63.5 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 \leftarrow
 .

63.6 Online Browsing

Dive into [PLASMA_dposv_Tile](#)

64 PLASMA_dpotrf

64.1 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.

64.2 C Bindings

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

64.3 Fortran Bindings

```
PLASMA_DPOTRF(INTEGER uplo, INTEGER N, DOUBLE PRECISION A, INTEGER LDA, INTEGER INFO)
```

64.4 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. $LDA \geq \max(1, N)$.

64.5 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

64.6 Online Browsing

Dive into [PLASMA_dpotrf](#)

65 PLASMA_dpotrf_Tile

65.1 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.

65.2 C Bindings

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

65.3 Fortran Bindings

```
PLASMA_DPOTRF_TILE(INTEGER uplo, INTEGER*4 A, INTEGER INFO)
```

65.4 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.

```

65.5 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 ↵
.

```

65.6 Online Browsing

Dive into [PLASMA_dpotrf_Tile](#)

66 PLASMA_dpotrs

66.1 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.

66.2 C Bindings

```

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

```

66.3 Fortran Bindings

```

PLASMA_DPOTRS(INTEGER uplo, INTEGER N, INTEGER NRHS, DOUBLE PRECISION A, INTEGER LDA, ↵
DOUBLE PRECISION B, INTEGER LDB, INTEGER INFO)

```

66.4 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	double* (IN) The triangular factor U or L from the Cholesky factorization $A = U^*T^*U$ or $A = L^*L$ \leftarrow **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. $LDB \geq \max(1, N)$.

66.5 Return Value:

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

66.6 Online Browsing

Dive into [PLASMA_dpotsr](#)

67 PLASMA_dpotsr_Tile

67.1 Purpose

PLASMA_dpotsr_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.

67.2 C Bindings

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

67.3 Fortran Bindings

```
PLASMA_DPOTRS_TILE(INTEGER uplo, INTEGER*4 A, INTEGER*4 B, INTEGER INFO)
```

67.4 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 $A = L*L \leftrightarrow **T$, computed by PLASMA_dpotrf.
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.

67.5 Return Value:

= 0: successful exit

67.6 Online Browsing

Dive into [PLASMA_dpotsr_Tile](#)

68 PLASMA_dsgesv

68.1 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.

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: $RNRM < N * XNRM * ANRM * EPS * BWD$ 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*).

Actually, in its current state (PLASMA 2.1.0), the test is slightly relaxed.

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

68.2 C Bindings

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

68.3 Fortran Bindings

```
PLASMA_DSGESV(INTEGER N, INTEGER NRHS, DOUBLE PRECISION A, INTEGER LDA, INTEGER LH, INTEGER IPIVH, DOUBLE PRECISION B, INTEGER LDB, DOUBLE PRECISION X, INTEGER LDX, INTEGER ITER, INTEGER INFO)
```

68.4 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> (IN) The N-by-N coefficient matrix A. This matrix is not modified.
LDA	<code>int</code> (IN) The leading dimension of the array A. $LDA \geq \max(1, N)$.
B	<code>double*</code> (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	<code>double*</code> (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

68.5 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.
```

68.6 Online Browsing

Dive into [PLASMA_dsgesv](#)

69 PLASMA_dsgesv_Tile

69.1 Purpose

PLASMA_dsgesv_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. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

PLASMA_dsgesv_Tile 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: $RNRM < N * XNRM * ANRM * EPS * BWD$ 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*).

Actually, in his current state (PLASMA 2.1.0), the test is slightly relaxed.

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

69.2 C Bindings

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

69.3 Fortran Bindings

```
PLASMA_DSGESV_TILE(INTEGER*4 A, INTEGER4 L, INTEGER IPIVH, INTEGER*4 B, INTEGER*4 X, ←
    INTEGER ITER, INTEGER INFO)
```

69.4 Arguments

A	<p>double* (In or INOUT)</p> <p>On entry, the N-by-N coefficient matrix A.</p> <ul style="list-style-type: none"> - if the iterative refinement converged, A is not modified; - otherwise, it failed backed to double precision solution, and then A contains the tile L and U factors from the factorization (not ← <p>equivalent to LAPACK).</p>
L	<p>double* (NODEP or OUT)</p> <p>On exit:</p> <ul style="list-style-type: none"> - if the iterative refinement converged, L is not modified; - otherwise, it failed backed to double precision solution, and then L is an auxiliary factorization data, related to the tile L factor,

necessary to solve the system of equations (not equivalent to LAPACK).

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

B `double*` (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.

69.5 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.

69.6 Online Browsing

Dive into [PLASMA_dsgesv_Tile](#)

70 PLASMA_dtrsm

70.1 Purpose

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

70.2 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)
```

70.3 Fortran Bindings

```
PLASMA_DTRSM(INTEGER side, INTEGER uplo, INTEGER transA, INTEGER diag, INTEGER N, INTEGER ↔
    NRHS, DOUBLE PRECISION A, INTEGER LDA, DOUBLE PRECISION B, INTEGER LDB, INTEGER INFO)
```

70.4 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.
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	double* (IN) The triangular matrix A. If uplo = PlasmaUpper, the leading N-by-N upper triangular \leftrightarrow 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	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. $LDB \geq \max(1, N)$.

70.5 Return Value:

```

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

```

70.6 Online Browsing

Dive into [PLASMA_dtrsm](#)

71 PLASMA_dtrsm_Tile

71.1 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.

71.2 C Bindings

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

71.3 Fortran Bindings

```
PLASMA_DTRSM_TILE(INTEGER side, INTEGER uplo, INTEGER transA, INTEGER diag, INTEGER*4 A, ←
INTEGER*4 B, INTEGER INFO)
```

71.4 Arguments

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

71.5 Return Value:

= 0: successful exit

71.6 Online Browsing

Dive into [PLASMA_dtrsm_Tile](#)

72 PLASMA_dtrsmpi

72.1 Purpose

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

72.2 C Bindings

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

72.3 Fortran Bindings

```
PLASMA_DTRSMPL(INTEGER N, INTEGER NRHS, DOUBLE PRECISION A, INTEGER LDA, INTEGER LH, ←
INTEGER IPIVH, DOUBLE PRECISION B, INTEGER LDB, INTEGER INFO)
```

72.4 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, if <code>return</code> value = 0, the N-by-NRHS solution matrix X.
LDB	<code>double*</code> (IN) The leading dimension of the array B. $LDB \geq \max(1, N)$.

72.5 Return Value:

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

72.6 Online Browsing

Dive into [PLASMA_dtrsimpl](#)

73 PLASMA_dtrsimpl_Tile

73.1 Purpose

PLASMA_dtrsimpl_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.

73.2 C Bindings

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

73.3 Fortran Bindings

```
PLASMA_DTRSMPL_TILE(INTEGER*4 A, INTEGER*4 L, INTEGER IPIVH, INTEGER*4 B, INTEGER INFO)
```

73.4 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, <code>if return</code> value = 0, the N-by-NRHS solution matrix X.

73.5 Return Value:

```
= 0: successful exit
```

73.6 Online Browsing

Dive into [PLASMA_dtrsimpl_Tile](#)

74 PLASMA_sgelqf

74.1 Purpose

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

74.2 C Bindings

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

74.3 Fortran Bindings

```
PLASMA_SGELQF(INTEGER M, INTEGER N, REAL A, INTEGER LDA, INTEGER T, INTEGER INFO)
```

74.4 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.

74.5 Return Value:

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

74.6 Online Browsing

Dive into [PLASMA_sgelqf](#)

75 PLASMA_sgelqf_Tile

75.1 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.

75.2 C Bindings

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

75.3 Fortran Bindings

```
PLASMA_SGELQF_TILE(INTEGER*4 A, INTEGER*4 T, INTEGER INFO)
```

75.4 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 if $M \leq N$); the elements above the diagonal represent the unitary matrix Q as a product of elementary reflectors, stored by tiles.</p>
T	<p><code>float*</code> (OUT)</p> <p>On exit, auxiliary factorization data, required by PLASMA_sgelqs to solve the system of equations.</p>

75.5 Return Value:

```
= 0: successful exit
```

75.6 Online Browsing

Dive into [PLASMA_sgelqf_Tile](#)

76 PLASMA_sgelqs

76.1 Purpose

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

76.2 C Bindings

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

76.3 Fortran Bindings

```
PLASMA_SGELQS(INTEGER M, INTEGER N, INTEGER NRHS, REAL A, INTEGER LDA, INTEGER T, REAL B,  $\leftarrow$ 
  INTEGER LDB, INTEGER INFO)
```


76.4 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 <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.
LDB	<code>int</code> (IN) The leading dimension of the array B. $LDB \geq N$.

76.5 Return Value:

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

76.6 Online Browsing

Dive into [PLASMA_sgelqs](#)

77 PLASMA_sgelqs_Tile

77.1 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.

77.2 C Bindings

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

77.3 Fortran Bindings

```
PLASMA_SGELQS_TILE(INTEGER*4 A, INTEGER*4 B, INTEGER*4 T, INTEGER INFO)
```

77.4 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.

77.5 Return Value:

`= 0`: successful exit

77.6 Online Browsing

Dive into [PLASMA_sgelqs_Tile](#)

78 PLASMA_sgels

78.1 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.

78.2 C Bindings

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

78.3 Fortran Bindings

```
PLASMA_SGELS(INTEGER trans, INTEGER M, INTEGER N, INTEGER NRHS, REAL A, INTEGER LDA,
INTEGER T, REAL B, INTEGER LDB, INTEGER INFO)
```

78.4 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	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.
LDA	int (IN) The leading dimension of the array A. $LDA \geq \max(1, M)$.
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 $M \geq N$, 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. $LDB \geq \max(1, M, N)$.

78.5 Return Value:

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

78.6 Online Browsing

Dive into [PLASMA_sgels](#)

79 PLASMA_sgels_Tile

79.1 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.

79.2 C Bindings

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

79.3 Fortran Bindings

```
PLASMA_SGELS_TILE(INTEGER trans, INTEGER*4 A, INTEGER*4 B, INTEGER*4 T, INTEGER INFO)
```

79.4 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 $M \geq N$, 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;

79.5 Return Value:

```
= 0: successful exit
```

79.6 Online Browsing

Dive into [PLASMA_sgels_Tile](#)

80 PLASMA_sgemmm

80.1 Purpose

PLASMA_sgemmm - Performs one of the matrix-matrix operations

$$C = \alpha * \text{op}(A) * \text{op}(B) + \beta * C,$$

where $\text{op}(X)$ is one of

$$\text{op}(X) = X \quad \text{or} \quad \text{op}(X) = X'$$

α and β are scalars, and A , B and C are matrices, with $\text{op}(A)$ an m by k matrix, $\text{op}(B)$ a k by n matrix and C an m by n matrix.

80.2 C Bindings

```
int PLASMA_sgemmm(PLASMA_enum transA, PLASMA_enum transB, int M, int N, int K, float alpha, ↵
    float *A, int LDA, float *B, int LDB, float beta, float *C, int LDC)
```

80.3 Fortran Bindings

```
PLASMA_SGEMM(INTEGER transA, INTEGER transB, INTEGER M, INTEGER N, INTEGER K, REAL alpha, ↵
    REAL A, INTEGER LDA, REAL B, INTEGER LDB, REAL beta, REAL C, INTEGER LDC, INTEGER INFO)
```

80.4 Arguments

```
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.
        Currently only PlasmaNoTrans is supported

transB  PLASMA_enum (IN)
        Specifies whether the matrix B is transposed, not transposed or conjugate ↵
        transposed:
        = PlasmaNoTrans:  B is transposed;
        = PlasmaTrans:   B is not transposed;
        = PlasmaTrans: B is conjugate transposed.
        Currently only PlasmaNoTrans is supported
```

M	<code>int</code> (IN) M specifies the number of rows of the matrix <code>op(A)</code> and of the matrix C. $M \geq 0$.
N	<code>int</code> (IN) N specifies the number of columns of the matrix <code>op(B)</code> and of the matrix C. $N \geq 0$. ↩
K	<code>int</code> (IN) K specifies the number of columns of the matrix <code>op(A)</code> and the number of rows of the matrix <code>op(B)</code> . $K \geq 0$.
alpha	<code>float</code> (IN) alpha specifies the scalar alpha
A	<code>float*</code> (IN) A is a LDA-by-ka matrix, where ka is K when <code>transA = PlasmaNoTrans</code> , and is M otherwise.
LDA	<code>int</code> (IN) The leading dimension of the array A. $LDA \geq \max(1, M)$.
B	<code>float*</code> (IN) B is a LDB-by-kb matrix, where kb is N when <code>transB = PlasmaNoTrans</code> , and is K otherwise.
LDB	<code>int</code> (IN) The leading dimension of the array B. $LDB \geq \max(1, N)$.
beta	<code>float</code> (IN) beta specifies the scalar beta
C	<code>float*</code> (INOUT) C is a LDC-by-N matrix. On exit, the array is overwritten by the M by N matrix $(\alpha * \text{op}(A) * \text{op}(B) + \beta * C)$ ↩
LDC	<code>int</code> (IN) The leading dimension of the array C. $LDC \geq \max(1, M)$.

80.5 Return Value:

= 0: successful exit

80.6 Online Browsing

Dive into [PLASMA_sgemm](#)

81 PLASMA_sgeqrf

81.1 Purpose

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

81.2 C Bindings

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

81.3 Fortran Bindings

```
PLASMA_SGEQRF(INTEGER M, INTEGER N, REAL A, INTEGER LDA, INTEGER T, INTEGER INFO)
```

81.4 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	<code>float*</code> (OUT) On exit, auxiliary factorization data, required by PLASMA_sgeqrs to solve the system of equations.

81.5 Return Value:

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

81.6 Online Browsing

Dive into [PLASMA_sgeqrf](#)

82 PLASMA_sgeqrf_Tile

82.1 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.

82.2 C Bindings

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

82.3 Fortran Bindings

```
PLASMA_SGEQRF_TILE(INTEGER*4 A, INTEGER*4 T, INTEGER INFO)
```

82.4 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 above the diagonal of the array contain the min(M,N)-by-N upper trapezoidal matrix R (R is upper triangular if <code>M >= N</code>); the elements below the diagonal represent the unitary matrix Q as a product of elementary reflectors stored by tiles.</p>
T	<p><code>float*</code> (OUT)</p> <p>On exit, auxiliary factorization data, required by PLASMA_sgeqrs to solve the system of equations.</p>

82.5 Return Value:

```
= 0: successful exit
```

82.6 Online Browsing

Dive into [PLASMA_sgeqrf_Tile](#)

83 PLASMA_sgeqrs

83.1 Purpose

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

83.2 C Bindings

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

83.3 Fortran Bindings

```
PLASMA_SGEQRS(INTEGER M, INTEGER N, INTEGER NRHS, REAL A, INTEGER LDA, INTEGER T, REAL B, INTEGER LDB, INTEGER INFO)
```


83.4 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>PLASMA_sgeqrf</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 <code>PLASMA_sgeqrf</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.
LDB	<code>int</code> (IN) The leading dimension of the array B. $LDB \geq \max(1, N)$.

83.5 Return Value:

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

83.6 Online Browsing

Dive into [PLASMA_sgeqrs](#)

84 PLASMA_sgeqrs_Tile

84.1 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.

84.2 C Bindings

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

84.3 Fortran Bindings

```
PLASMA_SGEQRS_TILE(INTEGER*4 A, INTEGER*4 B, INTEGER*4 T, INTEGER INFO)
```

84.4 Arguments

A	<code>float*</code> (INOUT) Details of the QR factorization of the original matrix A as returned by <code>PLASMA_sgeqrf</code> .
T	<code>float*</code> (IN) Auxiliary factorization data, computed by <code>PLASMA_sgeqrf</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.

84.5 Return Value:

= 0: successful exit

84.6 Online Browsing

Dive into [PLASMA_sgeqrs_Tile](#)

85 PLASMA_sgesv

85.1 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 $A * X = B$.

85.2 C Bindings

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

85.3 Fortran Bindings

```
PLASMA_SGESV(INTEGER N, INTEGER NRHS, REAL A, INTEGER LDA, INTEGER LH, INTEGER IPIVH, REAL ↵
B, INTEGER LDB, INTEGER INFO)
```

85.4 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 ↔).
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) ↔.
B	<code>float*</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.
LDB	<code>int</code> (IN) The leading dimension of the array B. $LDB \geq \max(1, N)$.

85.5 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.

```

85.6 Online Browsing

Dive into [PLASMA_sgesv](#)

86 PLASMA_sgesv_Tile

86.1 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.

86.2 C Bindings

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

86.3 Fortran Bindings

```
PLASMA_SGESV_TILE(INTEGER*4 A, INTEGER*4 L, INTEGER IPIVH, INTEGER*4 B, INTEGER INFO)
```

86.4 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.

86.5 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.
```

86.6 Online Browsing

Dive into [PLASMA_sgesv_Tile](#)

87 PLASMA_sgetrf

87.1 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.

87.2 C Bindings

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

87.3 Fortran Bindings

```
PLASMA_SGETRF(INTEGER M, INTEGER N, REAL A, INTEGER LDA, INTEGER LH, INTEGER IPIVH, INTEGER ↔  
INFO)
```

87.4 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 <code>PLASMA_sgetrs</code> to solve the system of equations.
IPIV	<code>int*</code> (OUT) The pivot indices that define the permutations (not equivalent to LAPACK).

87.5 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.

```

87.6 Online Browsing

Dive into [PLASMA_sgetrf](#)

88 PLASMA_sgetrf_Tile

88.1 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.

88.2 C Bindings

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

88.3 Fortran Bindings

```
PLASMA_SGETRF_TILE(INTEGER*4 A, INTEGER*4 L, INTEGER IPIVH, INTEGER INFO)
```

88.4 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).

88.5 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.

```

88.6 Online Browsing

Dive into [PLASMA_sgetrf_Tile](#)

89 PLASMA_sgetrs

89.1 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.

89.2 C Bindings

```

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

```

89.3 Fortran Bindings

```

PLASMA_SGETRS(INTEGER uplo, INTEGER N, INTEGER NRHS, REAL A, INTEGER LDA, INTEGER LH, ←
INTEGER IPIVH, REAL B, INTEGER LDB, INTEGER INFO)

```

89.4 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 <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 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. $LDB \geq \max(1, N)$.

89.5 Return Value:

```

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

```

89.6 Online Browsing

Dive into [PLASMA_sgetrs](#)

90 PLASMA_sgetrs_Tile

90.1 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.

90.2 C Bindings

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

90.3 Fortran Bindings

```
PLASMA_SGETRS_TILE(INTEGER*4 A, INTEGER*4 L, INTEGER IPIVH, INTEGER*4 B, INTEGER INFO)
```

90.4 Arguments

A	<code>float*</code> (IN) The tile factors L and U from the factorization, computed by <code>PLASMA_sgetrf</code> .
L	<code>float*</code> (IN) Auxiliary factorization data, related to the tile L factor, computed by <code>↔</code> <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 matrix of right hand side matrix B. On exit, the solution matrix X.

90.5 Return Value:

`= 0`: successful exit

90.6 Online Browsing

Dive into [PLASMA_sgetrs_Tile](#)

91 PLASMA_sposv

91.1 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 $A * X = B$.

91.2 C Bindings

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

91.3 Fortran Bindings

```
PLASMA_SPOSV(INTEGER uplo, INTEGER N, INTEGER NRHS, REAL A, INTEGER LDA, REAL B, INTEGER ↔  
LDB, INTEGER INFO)
```


91.4 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 \leftarrow ≥ 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 \leftarrow 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 \leftarrow 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. $LDA \geq \max(1, N)$.
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. $LDB \geq \max(1, N)$.

91.5 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  $\leftarrow$ 
      .

```

91.6 Online Browsing

Dive into [PLASMA_sposv](#)

92 PLASMA_sposv_Tile

92.1 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.

92.2 C Bindings

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

92.3 Fortran Bindings

```
PLASMA_SPOSV_TILE(INTEGER uplo, INTEGER*4 A, INTEGER*4 B, INTEGER INFO)
```

92.4 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 <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 $A = L^{**T}L$.
B	float* (INOUT) On entry, the N -by- $NRHS$ right hand side matrix B . On exit, if <code>return value = 0</code> , the N -by- $NRHS$ solution matrix X .

92.5 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
      .

```

92.6 Online Browsing

Dive into [PLASMA_sposv_Tile](#)

93 PLASMA_spotrf

93.1 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.

93.2 C Bindings

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

93.3 Fortran Bindings

```
PLASMA_SPOTRF(INTEGER uplo, INTEGER N, REAL A, INTEGER LDA, INTEGER INFO)
```

93.4 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	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. $LDA \geq \max(1, N)$.

93.5 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 ↩

93.6 Online Browsing

Dive into [PLASMA_spotrf](#)

94 PLASMA_spotrf_Tile

94.1 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.

94.2 C Bindings

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

94.3 Fortran Bindings

```
PLASMA_SPOTRF_TILE(INTEGER uplo, INTEGER*4 A, INTEGER INFO)
```

94.4 Arguments

uplo	PLASMA_enum (IN) = PlasmaUpper: Upper triangle of A is stored; = PlasmaLower: Lower triangle of A is stored.
A	<code>float*</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. 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 $A = L*L^*T$.

94.5 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
 .

94.6 Online Browsing

Dive into [PLASMA_spotrf_Tile](#)

95 PLASMA_spotrs

95.1 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**.

95.2 C Bindings

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

95.3 Fortran Bindings

```
PLASMA_SPOTRS(INTEGER uplo, INTEGER N, INTEGER NRHS, REAL A, INTEGER LDA, REAL B, INTEGER LDB, INTEGER INFO) ↩
```

95.4 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 $A = L*L**T$, 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. $LDB \geq \max(1, N)$.

95.5 Return Value:

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

95.6 Online Browsing

Dive into [PLASMA_spotrs](#)

96 PLASMA_spotrs_Tile

96.1 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.

96.2 C Bindings

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

96.3 Fortran Bindings

```
PLASMA_SPOTRS_TILE(INTEGER uplo, INTEGER*4 A, INTEGER*4 B, INTEGER INFO)
```

96.4 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 $A = L*L \leftrightarrow **T$, 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.

96.5 Return Value:

```
= 0: successful exit
```

96.6 Online Browsing

Dive into [PLASMA_spotrs_Tile](#)

97 PLASMA_strsm

97.1 Purpose

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

97.2 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)
```

97.3 Fortran Bindings

```
PLASMA_STRSM(INTEGER side, INTEGER uplo, INTEGER transA, INTEGER diag, INTEGER N, INTEGER ←
    NRHS, REAL A, INTEGER LDA, REAL B, INTEGER LDB, INTEGER INFO)
```

97.4 Arguments

side	PLASMA_enum (IN) Specifies whether A appears on the left or on the right of X: = PlasmaLeft: $A \cdot X = B$ = PlasmaRight: $X \cdot A = B$
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. $N \geq 0$.
NRHS	int (IN) The number of right hand sides, i.e., the number of columns of the matrix B. NRHS ← ≥ 0 .
A	float* (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	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. $LDB \geq \max(1, N)$.

97.5 Return Value:

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

97.6 Online Browsing

Dive into [PLASMA_strsm](#)

98 PLASMA_strsm_Tile

98.1 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.

98.2 C Bindings

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

98.3 Fortran Bindings

```
PLASMA_STRSM_TILE(INTEGER side, INTEGER uplo, INTEGER transA, INTEGER diag, INTEGER*4 A, ↵
INTEGER*4 B, INTEGER INFO)
```

98.4 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 us 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.

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.

```

98.5 Return Value:

```

= 0: successful exit

```

98.6 Online Browsing

Dive into [PLASMA_strsm_Tile](#)

99 PLASMA_strsmpl

99.1 Purpose

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

99.2 C Bindings

```

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

```

99.3 Fortran Bindings

```

PLASMA_STRSMPL(INTEGER N, INTEGER NRHS, REAL A, INTEGER LDA, INTEGER LH, INTEGER IPIV, ↔
REAL B, INTEGER LDB, INTEGER INFO)

```

99.4 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      float* (IN)

```

	The tile factor L 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 <code>PLASMA_sgetrf</code> .
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, if <code>return</code> value = 0, the N-by-NRHS solution matrix X.
LDB	<code>float*</code> (IN) The leading dimension of the array B. LDB $\geq \max(1,N)$.

99.5 Return Value:

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

99.6 Online Browsing

Dive into [PLASMA_strsmpl](#)

100 PLASMA_strsmpl_Tile

100.1 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.

100.2 C Bindings

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

100.3 Fortran Bindings

```
PLASMA_STRSMPL_TILE(INTEGER*4 A, INTEGER*4 L, INTEGER IPIV, INTEGER*4 B, INTEGER INFO)
```

100.4 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 `int*` (IN)
The pivot indices from `PLASMA_sgetrf` (not equivalent to LAPACK).

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.

100.5 Return Value:

= 0: successful exit

100.6 Online Browsing

Dive into [PLASMA_strsmpl_Tile](#)

101 PLASMA_Lapack_to_Tile

101.1 Purpose

PLASMA_Lapack_to_Tile - Conversion from LAPACK layout to tile layout.

101.2 C Bindings

```
int PLASMA_Lapack_to_Tile(void *Af77, int LDA, PLASMA_desc *A)
```

101.3 Fortran Bindings

```
PLASMA_LAPACK_TO_TILE(INTEGER*4 Af77, INTEGER LDA, INTEGER*4 A, INTEGER INFO)
```

101.4 Arguments

Af77 `void*` (IN)
LAPACK matrix.

LDA `int` (IN)
The leading dimension of the matrix Af77.

A `PLASMA_desc*` (OUT)
Descriptor of the PLASMA matrix in tile layout.

101.5 Return Value:

= `PLASMA_SUCCESS`: successful exit

101.6 Online Browsing

Dive into [PLASMA_Lapack_to_Tile](#)

102 PLASMA_Tile_to_Lapack

102.1 Purpose

PLASMA_Tile_to_Lapack - Conversion from tile layout to LAPACK layout

102.2 C Bindings

```
int PLASMA_Tile_to_Lapack(PLASMA_desc *A, void *Af77, int LDA)
```

102.3 Fortran Bindings

```
PLASMA_TILE_TO_LAPACK(INTEGER*4 A, INTEGER*4 Af77, INTEGER LDA, INTEGER INFO)
```

102.4 Arguments

A	PLASMA_desc* (OUT) Descriptor of the PLASMA matrix in tile layout.
Af77	void* (IN) LAPACK matrix.
LDA	int (IN) The leading dimension of the matrix Af77.

102.5 Return Value:

```
= PLASMA_SUCCESS: successful exit
```

102.6 Online Browsing

Dive into [PLASMA_Tile_to_Lapack](#)

103 PLASMA_Dealloc_Handle

103.1 Purpose

PLASMA_Dealloc_Handle - Deallocate workspace handle allocated by any workspace allocation routine.

103.2 C Bindings

103.3 Fortran Bindings

```
PLASMA_DEALLOC_HANDLE (INTEGER sp, INTEGER INFO)
```

103.4 Arguments

```
handle    void** (IN)  
          Workspace handle
```

103.5 Return Value:

```
= PLASMA_SUCCESS: successful exit
```

103.6 Online Browsing

Dive into [PLASMA_Dealloc_Handle](#)

104 PLASMA_Dealloc_Handle

104.1 Purpose

PLASMA_Dealloc_Handle - Deallocate workspace handle allocated by any workspace allocation routine.

104.2 C Bindings

104.3 Fortran Bindings

```
PLASMA_DEALLOC_HANDLE (INTEGER sp, INTEGER INFO)
```

104.4 Arguments

```
handle    void** (IN)  
          Workspace handle
```

104.5 Return Value:

```
= PLASMA_SUCCESS: successful exit
```

104.6 Online Browsing

Dive into [PLASMA_Dealloc_Handle](#)

105 PLASMA_Alloc_Workspace_cgels

105.1 Purpose

PLASMA_Alloc_Workspace_cgels - Allocates workspace for PLASMA_cgels or PLASMA_cgels_Tile routine.

105.2 C Bindings

```
int PLASMA_Alloc_Workspace_cgels(int M, int N, PLASMA_Complex32_t **T)
```

105.3 Fortran Bindings

```
PLASMA_ALLOC_WORKSPACE_CGELS(INTEGER M, INTEGER N, INTEGER T, INTEGER INFO)
```

105.4 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. $N \geq 0$.
T	<code>PLASMA_Complex32_t*</code> (OUT) On exit, workspace handle <code>for</code> storage of the extra T factors required by the tile \leftrightarrow QR or the tile LQ factorization.

105.5 Return Value:

```
= 0: successful exit
```

105.6 Online Browsing

Dive into [PLASMA_Alloc_Workspace_cgels](#)

106 PLASMA_Alloc_Workspace_cgeqrf

106.1 Purpose

PLASMA_Alloc_Workspace_cgeqrf - Allocates workspace for PLASMA_cgeqrf or PLASMA_cgeqrf_Tile routine.

106.2 C Bindings

```
int PLASMA_Alloc_Workspace_cgeqrf(int M, int N, PLASMA_Complex32_t **T)
```

106.3 Fortran Bindings

```
PLASMA_ALLOC_WORKSPACE_CGEQRF(INTEGER M, INTEGER N, INTEGER T, INTEGER INFO)
```

106.4 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. $N \geq 0$.
T	<code>PLASMA_Complex32_t*</code> (OUT) On exit, workspace handle <code>for</code> storage of the extra T factors required by the tile QR factorization. \leftrightarrow

106.5 Return Value:

```
= 0: successful exit
```

106.6 Online Browsing

Dive into [PLASMA_Alloc_Workspace_cgeqrf](#)

107 PLASMA_Alloc_Workspace_cgelqf

107.1 Purpose

PLASMA_Alloc_Workspace_cgelqf - Allocates workspace for `PLASMA_cgelqf` or `PLASMA_cgelqf_Tile` routines.

107.2 C Bindings

```
int PLASMA_Alloc_Workspace_cgelqf(int M, int N, PLASMA_Complex32_t **T)
```

107.3 Fortran Bindings

```
PLASMA_ALLOC_WORKSPACE_CGELQF(INTEGER M, INTEGER N, INTEGER T, INTEGER INFO)
```

107.4 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. $N \geq 0$.
T	<code>PLASMA_Complex32_t*</code> (OUT)

On exit, workspace handle `for` storage of the extra T factors required by the tile LU factorization. ↩

107.5 Return Value:

= 0: successful exit

107.6 Online Browsing

Dive into [PLASMA_Alloc_Workspace_cgelqf](#)

108 PLASMA_Alloc_Workspace_cgesv

108.1 Purpose

PLASMA_Alloc_Workspace_cgesv - Allocates workspace for PLASMA_cgesv or PLASMA_cgesv_Tile routines.

108.2 C Bindings

```
int PLASMA_Alloc_Workspace_cgesv(int N, PLASMA_Complex32_t **L, int **IPIV)
```

108.3 Fortran Bindings

```
PLASMA_ALLOC_WORKSPACE_CGESV(INTEGER N, INTEGER L, INTEGER IPIV, INTEGER INFO)
```

108.4 Arguments

N	<code>int</code> (IN) The number of linear equations, i.e., the order of the matrix A. N >= 0.
L	<code>PLASMA_Complex32_t*</code> (OUT) On exit, workspace handle <code>for</code> storage of the extra L factors required by the tile LU factorization. ↩
IPIV	<code>int**</code> (OUT) On exit, workspace handle <code>for</code> storage of pivot indexes required by the tile LU factorization (not equivalent to LAPACK).

108.5 Return Value:

= 0: successful exit

108.6 Online Browsing

Dive into [PLASMA_Alloc_Workspace_cgesv](#)

109 PLASMA_Alloc_Workspace_cgetrf

109.1 Purpose

PLASMA_Alloc_Workspace_cgetrf - Allocates workspace for PLASMA_cgetrf or PLASMA_cgetrf_Tile routines.

109.2 C Bindings

```
int PLASMA_Alloc_Workspace_cgetrf(int M, int N, PLASMA_Complex32_t **L, int **IPIV)
```

109.3 Fortran Bindings

```
PLASMA_ALLOC_WORKSPACE_CGETRF(INTEGER M, INTEGER N, INTEGER L, INTEGER IPIV, INTEGER INFO)
```

109.4 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. $N \geq 0$.
L	<code>PLASMA_Complex32_t*</code> (OUT) On exit, workspace handle <code>for</code> storage of the extra L factors required by the tile LU factorization. ↔
IPIV	<code>int**</code> (OUT) On exit, workspace handle <code>for</code> storage of pivot indexes required by the tile LU factorization (not equivalent to LAPACK).

109.5 Return Value:

```
= 0: successful exit
```

109.6 Online Browsing

Dive into [PLASMA_Alloc_Workspace_cgetrf](#)

110 PLASMA_Alloc_Workspace_dgels

110.1 Purpose

PLASMA_Alloc_Workspace_dgels - Allocates workspace for PLASMA_dgels or PLASMA_dgels_Tile routine.

110.2 C Bindings

```
int PLASMA_Alloc_Workspace_dgels(int M, int N, double **T)
```

110.3 Fortran Bindings

```
PLASMA_ALLOC_WORKSPACE_DGELS(INTEGER M, INTEGER N, INTEGER T, INTEGER INFO)
```

110.4 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. $N \geq 0$.
T	<code>double*</code> (OUT) On exit, workspace handle <code>for</code> storage of the extra T factors required by the tile \leftrightarrow QR or the tile LQ factorization.

110.5 Return Value:

```
= 0: successful exit
```

110.6 Online Browsing

Dive into [PLASMA_Alloc_Workspace_dgels](#)

111 PLASMA_Alloc_Workspace_dgeqrf

111.1 Purpose

PLASMA_Alloc_Workspace_dgeqrf - Allocates workspace for PLASMA_dgeqrf or PLASMA_dgeqrf_Tile routine.

111.2 C Bindings

```
int PLASMA_Alloc_Workspace_dgeqrf(int M, int N, double **T)
```

111.3 Fortran Bindings

```
PLASMA_ALLOC_WORKSPACE_DGEQRF(INTEGER M, INTEGER N, INTEGER T, INTEGER INFO)
```

111.4 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. $N \geq 0$.
T	<code>double*</code> (OUT)

On exit, workspace handle `for` storage of the extra T factors required by the tile \leftrightarrow
 QR
 factorization.

111.5 Return Value:

= 0: successful exit

111.6 Online Browsing

Dive into [PLASMA_Alloc_Workspace_dgeqrf](#)

112 PLASMA_Alloc_Workspace_dgelqf

112.1 Purpose

PLASMA_Alloc_Workspace_dgelqf - Allocates workspace for PLASMA_dgelqf or PLASMA_dgelqf_Tile routines.

112.2 C Bindings

```
int PLASMA_Alloc_Workspace_dgelqf(int M, int N, double **T)
```

112.3 Fortran Bindings

```
PLASMA_ALLOC_WORKSPACE_DGELQF(INTEGER M, INTEGER N, INTEGER T, INTEGER INFO)
```

112.4 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. $N \geq 0$.
T	<code>double*</code> (OUT) On exit, workspace handle <code>for</code> storage of the extra T factors required by the tile \leftrightarrow LQ factorization.

112.5 Return Value:

= 0: successful exit

112.6 Online Browsing

Dive into [PLASMA_Alloc_Workspace_dgelqf](#)

113 PLASMA_Alloc_Workspace_dgesv

113.1 Purpose

PLASMA_Alloc_Workspace_dgesv - Allocates workspace for PLASMA_dgesv or PLASMA_dgesv_Tile routines.

113.2 C Bindings

```
int PLASMA_Alloc_Workspace_dgesv(int N, double **L, int **IPIV)
```

113.3 Fortran Bindings

```
PLASMA_ALLOC_WORKSPACE_DGESV(INTEGER N, INTEGER L, INTEGER IPIV, INTEGER INFO)
```

113.4 Arguments

N	<code>int</code> (IN) The number of linear equations, i.e., the order of the matrix A. $N \geq 0$.
L	<code>double*</code> (OUT) On exit, workspace handle <code>for</code> storage of the extra L factors required by the tile LU factorization. ↔
IPIV	<code>int**</code> (OUT) On exit, workspace handle <code>for</code> storage of pivot indexes required by the tile LU factorization (not equivalent to LAPACK).

113.5 Return Value:

```
= 0: successful exit
```

113.6 Online Browsing

Dive into [PLASMA_Alloc_Workspace_dgesv](#)

114 PLASMA_Alloc_Workspace_dgetrf

114.1 Purpose

PLASMA_Alloc_Workspace_dgetrf - Allocates workspace for PLASMA_dgetrf or PLASMA_dgetrf_Tile routines.

114.2 C Bindings

```
int PLASMA_Alloc_Workspace_dgetrf(int M, int N, double **L, int **IPIV)
```

114.3 Fortran Bindings

```
PLASMA_ALLOC_WORKSPACE_DGETRF(INTEGER M, INTEGER N, INTEGER L, INTEGER IPIV, INTEGER INFO)
```

114.4 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. $N \geq 0$.
L	<code>double*</code> (OUT) On exit, workspace handle <code>for</code> storage of the extra L factors required by the tile LU factorization. \leftrightarrow
IPIV	<code>int**</code> (OUT) On exit, workspace handle <code>for</code> storage of pivot indexes required by the tile LU factorization (not equivalent to LAPACK).

114.5 Return Value:

```
= 0: successful exit
```

114.6 Online Browsing

Dive into [PLASMA_Alloc_Workspace_dgetrf](#)

115 PLASMA_Alloc_Workspace_sgels

115.1 Purpose

PLASMA_Alloc_Workspace_sgels - Allocates workspace for **PLASMA_sgels** or **PLASMA_sgels_Tile** routine.

115.2 C Bindings

```
int PLASMA_Alloc_Workspace_sgels(int M, int N, float **T)
```

115.3 Fortran Bindings

```
PLASMA_ALLOC_WORKSPACE_SGELS(INTEGER M, INTEGER N, INTEGER T, INTEGER INFO)
```

115.4 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. $N \geq 0$.
T	<code>float*</code> (OUT) On exit, workspace handle <code>for</code> storage of the extra T factors required by the tile \leftrightarrow QR or the tile LQ factorization.

115.5 Return Value:

`= 0`: successful exit

115.6 Online Browsing

Dive into [PLASMA_Alloc_Workspace_sgels](#)

116 PLASMA_Alloc_Workspace_sgeqrf

116.1 Purpose

PLASMA_Alloc_Workspace_sgeqrf - Allocates workspace for PLASMA_sgeqrf or PLASMA_sgeqrf_Tile routine.

116.2 C Bindings

```
int PLASMA_Alloc_Workspace_sgeqrf(int M, int N, float **T)
```

116.3 Fortran Bindings

```
PLASMA_ALLOC_WORKSPACE_SGEQRF(INTEGER M, INTEGER N, INTEGER T, INTEGER INFO)
```

116.4 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. $N \geq 0$.
T	<code>float*</code> (OUT) On exit, workspace handle <code>for</code> storage of the extra T factors required by the tile \leftrightarrow QR factorization.

116.5 Return Value:

```
= 0: successful exit
```

116.6 Online Browsing

Dive into [PLASMA_Alloc_Workspace_sgeqrf](#)

117 PLASMA_Alloc_Workspace_sgelqf

117.1 Purpose

PLASMA_Alloc_Workspace_sgelqf - Allocates workspace for PLASMA_sgelqf or PLASMA_sgelqf_Tile routines.

117.2 C Bindings

```
int PLASMA_Alloc_Workspace_sgelqf(int M, int N, float **T)
```

117.3 Fortran Bindings

```
PLASMA_ALLOC_WORKSPACE_SGELQF(INTEGER M, INTEGER N, INTEGER T, INTEGER INFO)
```

117.4 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. $N \geq 0$.
T	<code>float*</code> (OUT) On exit, workspace handle <code>for</code> storage of the extra T factors required by the tile \leftrightarrow LQ factorization.

117.5 Return Value:

```
= 0: successful exit
```

117.6 Online Browsing

Dive into [PLASMA_Alloc_Workspace_sgelqf](#)

118 PLASMA_Alloc_Workspace_sgesv

118.1 Purpose

PLASMA_Alloc_Workspace_sgesv - Allocates workspace for PLASMA_sgesv or PLASMA_sgesv_Tile routines.

118.2 C Bindings

```
int PLASMA_Alloc_Workspace_sgesv(int N, float **L, int **IPIV)
```

118.3 Fortran Bindings

```
PLASMA_ALLOC_WORKSPACE_SGESV(INTEGER N, INTEGER L, INTEGER IPIV, INTEGER INFO)
```

118.4 Arguments

N	<code>int</code> (IN) The number of linear equations, i.e., the order of the matrix A. $N \geq 0$.
L	<code>float*</code> (OUT) On exit, workspace handle <code>for</code> storage of the extra L factors required by the tile LU factorization. \leftrightarrow
IPIV	<code>int**</code> (OUT) On exit, workspace handle <code>for</code> storage of pivot indexes required by the tile LU factorization (not equivalent to LAPACK).

118.5 Return Value:

```
= 0: successful exit
```

118.6 Online Browsing

Dive into [PLASMA_Alloc_Workspace_sgesv](#)

119 PLASMA_Alloc_Workspace_sgetrf

119.1 Purpose

PLASMA_Alloc_Workspace_sgetrf - Allocates workspace for PLASMA_sgetrf or PLASMA_sgetrf_Tile routines.

119.2 C Bindings

```
int PLASMA_Alloc_Workspace_sgetrf(int M, int N, float **L, int **IPIV)
```

119.3 Fortran Bindings

```
PLASMA_ALLOC_WORKSPACE_SGETRF(INTEGER M, INTEGER N, INTEGER L, INTEGER IPIV, INTEGER INFO)
```


119.4 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. $N \geq 0$.
L	<code>float*</code> (OUT) On exit, workspace handle <code>for</code> storage of the extra L factors required by the tile LU factorization. \leftrightarrow
IPIV	<code>int**</code> (OUT) On exit, workspace handle <code>for</code> storage of pivot indexes required by the tile LU factorization (not equivalent to LAPACK).

119.5 Return Value:

`= 0`: successful exit

119.6 Online Browsing

Dive into [PLASMA_Alloc_Workspace_sgetrf](#)

120 PLASMA_Alloc_Workspace_zgels

120.1 Purpose

PLASMA_Alloc_Workspace_zgels - Allocates workspace for **PLASMA_zgels** or **PLASMA_zgels_Tile** routine.

120.2 C Bindings

```
int PLASMA_Alloc_Workspace_zgels(int M, int N, PLASMA_Complex64_t **T)
```

120.3 Fortran Bindings

```
PLASMA_ALLOC_WORKSPACE_ZGELS(INTEGER M, INTEGER N, INTEGER T, INTEGER INFO)
```

120.4 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. $N \geq 0$.
T	<code>PLASMA_Complex64_t*</code> (OUT) On exit, workspace handle <code>for</code> storage of the extra T factors required by the tile QR or the tile LQ factorization. \leftrightarrow

120.5 Return Value:

```
= 0: successful exit
```

120.6 Online Browsing

Dive into [PLASMA_Alloc_Workspace_zgels](#)

121 PLASMA_Alloc_Workspace_zgeqrf

121.1 Purpose

PLASMA_Alloc_Workspace_zgeqrf - Allocates workspace for PLASMA_zgeqrf or PLASMA_zgeqrf_Tile routine.

121.2 C Bindings

```
int PLASMA_Alloc_Workspace_zgeqrf(int M, int N, PLASMA_Complex64_t **T)
```

121.3 Fortran Bindings

```
PLASMA_ALLOC_WORKSPACE_ZGEQRF(INTEGER M, INTEGER N, INTEGER T, INTEGER INFO)
```

121.4 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. $N \geq 0$.
T	<code>PLASMA_Complex64_t*</code> (OUT) On exit, workspace handle <code>for</code> storage of the extra T factors required by the tile QR factorization. \leftrightarrow

121.5 Return Value:

```
= 0: successful exit
```

121.6 Online Browsing

Dive into [PLASMA_Alloc_Workspace_zgeqrf](#)

122 PLASMA_Alloc_Workspace_zgelqf

122.1 Purpose

PLASMA_Alloc_Workspace_zgelqf - Allocates workspace for PLASMA_zgelqf or PLASMA_zgelqf_Tile routines.

122.2 C Bindings

```
int PLASMA_Alloc_Workspace_zgelqf(int M, int N, PLASMA_Complex64_t **T)
```

122.3 Fortran Bindings

```
PLASMA_ALLOC_WORKSPACE_ZGELQF(INTEGER M, INTEGER N, INTEGER T, INTEGER INFO)
```

122.4 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. $N \geq 0$.
T	<code>PLASMA_Complex64_t*</code> (OUT) On exit, workspace handle <code>for</code> storage of the extra T factors required by the tile LQ factorization. ↔

122.5 Return Value:

```
= 0: successful exit
```

122.6 Online Browsing

Dive into [PLASMA_Alloc_Workspace_zgelqf](#)

123 PLASMA_Alloc_Workspace_zgesv

123.1 Purpose

PLASMA_Alloc_Workspace_zgesv - Allocates workspace for `PLASMA_zgesv` or `PLASMA_zgesv_Tile` routines.

123.2 C Bindings

```
int PLASMA_Alloc_Workspace_zgesv(int N, PLASMA_Complex64_t **L, int **IPIV)
```

123.3 Fortran Bindings

```
PLASMA_ALLOC_WORKSPACE_ZGESV(INTEGER N, INTEGER L, INTEGER IPIV, INTEGER INFO)
```

123.4 Arguments

N	<code>int</code> (IN) The number of linear equations, i.e., the order of the matrix A. $N \geq 0$.
L	<code>PLASMA_Complex64_t*</code> (OUT) On exit, workspace handle <code>for</code> storage of the extra L factors required by the tile LU factorization. \leftrightarrow
IPIV	<code>int**</code> (OUT) On exit, workspace handle <code>for</code> storage of pivot indexes required by the tile LU factorization (not equivalent to LAPACK).

123.5 Return Value:

`= 0`: successful exit

123.6 Online Browsing

Dive into [PLASMA_Alloc_Workspace_zgesv](#)

124 PLASMA_Alloc_Workspace_zgetrf

124.1 Purpose

PLASMA_Alloc_Workspace_zgetrf - Allocates workspace for `PLASMA_zgetrf` or `PLASMA_zgetrf_Tile` routines.

124.2 C Bindings

```
int PLASMA_Alloc_Workspace_zgetrf(int M, int N, PLASMA_Complex64_t **L, int **IPIV)
```

124.3 Fortran Bindings

```
PLASMA_ALLOC_WORKSPACE_ZGETRF(INTEGER M, INTEGER N, INTEGER L, INTEGER IPIV, INTEGER INFO)
```

124.4 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. $N \geq 0$.
L	<code>PLASMA_Complex64_t*</code> (OUT) On exit, workspace handle <code>for</code> storage of the extra L factors required by the tile LU factorization. \leftrightarrow
IPIV	<code>int**</code> (OUT) On exit, workspace handle <code>for</code> storage of pivot indexes required by the tile LU factorization (not equivalent to LAPACK).

124.5 Return Value:

```
= 0: successful exit
```

124.6 Online Browsing

Dive into [PLASMA_Alloc_Workspace_zgetrf](#)

125 PLASMA_zcgesv

125.1 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.

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: $RNRM < N * XNRM * ANRM * EPS * BWDMA$ 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*).

Actually, in its current state (PLASMA 2.1.0), the test is slightly relaxed.

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

125.2 C Bindings

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

125.3 Fortran Bindings

```
PLASMA_ZCGESV(INTEGER N, INTEGER NRHS, COMPLEX*16 A, INTEGER LDA, INTEGER LH, INTEGER IPIVH ←
    , COMPLEX*16 B, INTEGER LDB, COMPLEX*16 X, INTEGER LDX, INTEGER ITER, INTEGER INFO)
```

125.4 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> (IN) The N-by-N coefficient matrix A. This matrix is not modified.
LDA	<code>int</code> (IN) The leading dimension of the array A. $LDA \geq \max(1, N)$.
B	<code>PLASMA_Complex64_t*</code> (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	<code>PLASMA_Complex64_t*</code> (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 \leftrightarrow

125.5 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.

```

125.6 Online Browsing

Dive into [PLASMA_zcgesv](#)

126 PLASMA_zcgesv_Tile

126.1 Purpose

PLASMA_zcgesv_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. All matrices are passed through descriptors. All dimensions are taken from the descriptors.

PLASMA_zcgesv_Tile 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: $RNRM < N * XNRM * ANRM * EPS * BWDMA$ 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)`.

Actually, in his current state (PLASMA 2.1.0), the test is slightly relaxed.

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

126.2 C Bindings

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

126.3 Fortran Bindings

```
PLASMA_ZCGESV_TILE(INTEGER*4 A, INTEGER4 L, INTEGER IPIVH, INTEGER*4 B, INTEGER*4 X, ←
    INTEGER INFO)
```

126.4 Arguments

A	PLASMA_Complex64_t* (In or INOUT) On entry, the N-by-N coefficient matrix A. - if the iterative refinement converged, A is not modified; - otherwise, it failed backed to double precision solution, and then A contains the tile L and U factors from the factorization (not ← equivalent to LAPACK).
L	PLASMA_Complex64_t* (NODEP or OUT) On exit: - if the iterative refinement converged, L is not modified; - otherwise, it failed backed to double precision solution, and then L is an auxiliary factorization data, related to the tile L factor, necessary to solve the system of equations (not equivalent to LAPACK).
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.

126.5 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.
```

126.6 Online Browsing

Dive into [PLASMA_zcgesv_Tile](#)

127 PLASMA_zgelqf

127.1 Purpose

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

127.2 C Bindings

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

127.3 Fortran Bindings

```
PLASMA_ZGELQF(INTEGER M, INTEGER N, COMPLEX*16 A, INTEGER LDA, INTEGER T, INTEGER INFO)
```

127.4 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 <code>if</code> $M \leq N$); 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.

127.5 Return Value:

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


127.6 Online Browsing

Dive into [PLASMA_zgelqf](#)

128 PLASMA_zgelqf_Tile

128.1 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.

128.2 C Bindings

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

128.3 Fortran Bindings

```
PLASMA_ZGELQF_TILE(INTEGER*4 A, INTEGER*4 T, INTEGER INFO)
```

128.4 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(\leftarrow M,N) lower trapezoidal matrix L (L is lower triangular if $M \leq N$); the elements above \leftarrow the diagonal represent the unitary matrix Q as a product of elementary reflectors, \leftarrow stored by tiles.
T	PLASMA_Complex64_t* (OUT) On exit, auxiliary factorization data, required by PLASMA_zgelqs to solve the \leftarrow system of equations.

128.5 Return Value:

```
= 0: successful exit
```

128.6 Online Browsing

Dive into [PLASMA_zgelqf_Tile](#)

129 PLASMA_zgelqs

129.1 Purpose

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

129.2 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)
```

129.3 Fortran Bindings

```
PLASMA_ZGELQS(INTEGER M, INTEGER N, INTEGER NRHS, COMPLEX*16 A, INTEGER LDA, INTEGER T, ↔
    COMPLEX*16 B, INTEGER LDB, INTEGER INFO)
```

129.4 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. $LDB \geq N$.

129.5 Return Value:

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

129.6 Online Browsing

Dive into [PLASMA_zgelqs](#)

130 PLASMA_zgelqs_Tile

130.1 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.

130.2 C Bindings

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

130.3 Fortran Bindings

```
PLASMA_ZGELQS_TILE(INTEGER*4 A, INTEGER*4 B, INTEGER*4 T, INTEGER INFO)
```

130.4 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* (INOUT) On entry, the M-by-NRHS right hand side matrix B. On exit, the N-by-NRHS solution matrix X.

130.5 Return Value:

```
= 0: successful exit
```

130.6 Online Browsing

Dive into [PLASMA_zgelqs_Tile](#)

131 PLASMA_zgels

131.1 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 >= 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.

131.2 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)
```

131.3 Fortran Bindings

```
PLASMA_ZGELS(INTEGER trans, INTEGER M, INTEGER N, INTEGER NRHS, COMPLEX*16 A, INTEGER LDA, ←
             INTEGER T, COMPLEX*16 B, INTEGER LDB, INTEGER INFO)
```

131.4 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.
M	int (IN) The number of rows of the matrix A. M >= 0.
N	int (IN) The number of columns of the matrix A. N >= 0.
NRHS	int (IN) The number of right hand sides, i.e., the number of columns of the matrices B and ← X. NRHS >= 0.
A	PLASMA_Complex64_t* (INOUT) On entry, the M-by-N matrix A. On exit, if M >= 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.
LDA	int (IN) The leading dimension of the array A. LDA >= max(1,M).
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 >= 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;
LDB	int (IN) The leading dimension of the array B. LDB >= MAX(1,M,N).

131.5 Return Value:

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

131.6 Online Browsing

Dive into [PLASMA_zgels](#)

132 PLASMA_zgels_Tile

132.1 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 $M \geq N$: 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.

132.2 C Bindings

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

132.3 Fortran Bindings

```
PLASMA_ZGELS_TILE(INTEGER trans, INTEGER*4 A, INTEGER*4 B, INTEGER*4 T, INTEGER INFO)
```

132.4 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 <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;

```

132.5 Return Value:

```
= 0: successful exit
```

132.6 Online Browsing

Dive into [PLASMA_zgels_Tile](#)

133 PLASMA_zgemm

133.1 Purpose

PLASMA_zgemm - Performs one of the matrix-matrix operations

$$C = \alpha * \text{op}(A) * \text{op}(B) + \beta * C,$$

where $\text{op}(X)$ is one of

$$\text{op}(X) = X \quad \text{or} \quad \text{op}(X) = X'$$

α and β are scalars, and A, B and C are matrices, with $\text{op}(A)$ an m by k matrix, $\text{op}(B)$ a k by n matrix and C an m by n matrix.

133.2 C Bindings

```

int PLASMA_zgemm(PLASMA_enum transA, PLASMA_enum transB, int M, int N, int K, ↵
    PLASMA_Complex64_t alpha, PLASMA_Complex64_t *A, int LDA, PLASMA_Complex64_t *B, int LDB ↵
    , PLASMA_Complex64_t beta, PLASMA_Complex64_t *C, int LDC)

```

133.3 Fortran Bindings

```

PLASMA_ZGEMM(INTEGER transA, INTEGER transB, INTEGER M, INTEGER N, INTEGER K, COMPLEX*16 ↵
    alpha, COMPLEX*16 A, INTEGER LDA, COMPLEX*16 B, INTEGER LDB, COMPLEX*16 beta, COMPLEX*16 ↵
    C, INTEGER LDC, INTEGER INFO)

```

133.4 Arguments

transA	<p>PLASMA_enum (IN)</p> <p>Specifies whether the matrix A is transposed, not transposed or conjugate transposed: ↵</p> <p>= PlasmaNoTrans: A is transposed;</p> <p>= PlasmaTrans: A is not transposed;</p> <p>= PlasmaConjTrans: A is conjugate transposed.</p> <p>Currently only PlasmaNoTrans is supported</p>
transB	<p>PLASMA_enum (IN)</p> <p>Specifies whether the matrix B is transposed, not transposed or conjugate transposed: ↵</p> <p>= PlasmaNoTrans: B is transposed;</p> <p>= PlasmaTrans: B is not transposed;</p> <p>= PlasmaConjTrans: B is conjugate transposed.</p> <p>Currently only PlasmaNoTrans is supported</p>
M	<p>int (IN)</p> <p>M specifies the number of rows of the matrix op(A) and of the matrix C. $M \geq 0$.</p>
N	<p>int (IN)</p> <p>N specifies the number of columns of the matrix op(B) and of the matrix C. $N \geq 0$. ↵</p>
K	<p>int (IN)</p> <p>K specifies the number of columns of the matrix op(A) and the number of rows of the matrix op(B). $K \geq 0$.</p>
alpha	<p>PLASMA_Complex64_t (IN)</p> <p>alpha specifies the scalar alpha</p>
A	<p>PLASMA_Complex64_t* (IN)</p> <p>A is a LDA-by-ka matrix, where ka is K when transA = PlasmaNoTrans, and is M otherwise.</p>
LDA	<p>int (IN)</p> <p>The leading dimension of the array A. $LDA \geq \max(1, M)$.</p>
B	<p>PLASMA_Complex64_t* (IN)</p> <p>B is a LDB-by-kb matrix, where kb is N when transB = PlasmaNoTrans, and is K otherwise.</p>
LDB	<p>int (IN)</p> <p>The leading dimension of the array B. $LDB \geq \max(1, N)$.</p>
beta	<p>PLASMA_Complex64_t (IN)</p> <p>beta specifies the scalar beta</p>
C	<p>PLASMA_Complex64_t* (INOUT)</p> <p>C is a LDC-by-N matrix.</p> <p>On exit, the array is overwritten by the M by N matrix (alpha*op(A)*op(B) + ↵ beta*C)</p>
LDC	<p>int (IN)</p> <p>The leading dimension of the array C. $LDC \geq \max(1, M)$.</p>

133.5 Return Value:

= 0: successful exit

133.6 Online Browsing

Dive into [PLASMA_zgemm](#)

134 PLASMA_zgeqrf

134.1 Purpose

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

134.2 C Bindings

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

134.3 Fortran Bindings

```
PLASMA_ZGEQRF(INTEGER M, INTEGER N, COMPLEX*16 A, INTEGER LDA, INTEGER T, INTEGER INFO)
```

134.4 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 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>PLASMA_Complex64_t*</code> (OUT) On exit, auxiliary factorization data, required by <code>PLASMA_zgeqrs</code> to solve the system of equations.

134.5 Return Value:

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


134.6 Online Browsing

Dive into [PLASMA_zgeqrf](#)

135 PLASMA_zgeqrf_Tile

135.1 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.

135.2 C Bindings

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

135.3 Fortran Bindings

```
PLASMA_ZGEQRF_TILE(INTEGER*4 A, INTEGER*4 T, INTEGER INFO)
```

135.4 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.

135.5 Return Value:

```
= 0: successful exit
```

135.6 Online Browsing

Dive into [PLASMA_zgeqrf_Tile](#)

136 PLASMA_zgeqrs

136.1 Purpose

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

136.2 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)
```

136.3 Fortran Bindings

```
PLASMA_ZGEQRS(INTEGER M, INTEGER N, INTEGER NRHS, COMPLEX*16 A, INTEGER LDA, INTEGER T, ↵
    COMPLEX*16 B, INTEGER LDB, INTEGER INFO)
```

136.4 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> (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	<code>PLASMA_Complex64_t*</code> (IN) Auxiliary factorization data, computed by <code>PLASMA_zgeqrf</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. $LDB \geq \max(1, N)$.

136.5 Return Value:

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

136.6 Online Browsing

Dive into [PLASMA_zgeqrs](#)

137 PLASMA_zgeqrs_Tile

137.1 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.

137.2 C Bindings

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

137.3 Fortran Bindings

```
PLASMA_ZGEQRS_TILE(INTEGER*4 A, INTEGER*4 B, INTEGER*4 T, INTEGER INFO)
```

137.4 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.

137.5 Return Value:

```
= 0: successful exit
```

137.6 Online Browsing

Dive into [PLASMA_zgeqrs_Tile](#)

138 PLASMA_zgesv

138.1 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 $A * X = B$.

138.2 C Bindings

```
int PLASMA_zgesv(int N, int NRHS, PLASMA_Complex64_t *A, int LDA, PLASMA_Complex64_t *L, ←
    int *IPIV, PLASMA_Complex64_t *B, int LDB)
```

138.3 Fortran Bindings

```
PLASMA_ZGESV(INTEGER N, INTEGER NRHS, COMPLEX*16 A, INTEGER LDA, INTEGER LH, INTEGER IPIVH, ←
    COMPLEX*16 B, INTEGER LDB, INTEGER INFO)
```

138.4 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	<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. $LDB \geq \max(1, N)$.

138.5 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.

```

138.6 Online Browsing

Dive into [PLASMA_zgesv](#)

139 PLASMA_zgesv_Tile

139.1 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.

139.2 C Bindings

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

139.3 Fortran Bindings

```
PLASMA_ZGESV_TILE(INTEGER*4 A, INTEGER*4 L, INTEGER IPIVH, INTEGER*4 B, INTEGER INFO)
```

139.4 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.

139.5 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.
```

139.6 Online Browsing

Dive into [PLASMA_zgesv_Tile](#)

140 PLASMA_zgetrf

140.1 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.

140.2 C Bindings

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

140.3 Fortran Bindings

```
PLASMA_ZGETRF(INTEGER M, INTEGER N, COMPLEX*16 A, INTEGER LDA, INTEGER LH, INTEGER IPIVH, ←
              INTEGER INFO)
```

140.4 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).

140.5 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.
```

140.6 Online Browsing

Dive into [PLASMA_zgetrf](#)

141 PLASMA_zgetrf_Tile

141.1 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.

141.2 C Bindings

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

141.3 Fortran Bindings

```
PLASMA_ZGETRF_TILE(INTEGER*4 A, INTEGER*4 L, INTEGER IPIVH, INTEGER INFO)
```

141.4 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	<code>int*</code> (OUT) The pivot indices that define the permutations (not equivalent to LAPACK).

141.5 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.
```

141.6 Online Browsing

Dive into [PLASMA_zgetrf_Tile](#)

142 PLASMA_zgetrs

142.1 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.

142.2 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)
```

142.3 Fortran Bindings

```
PLASMA_ZGETRS(INTEGER uplo, INTEGER N, INTEGER NRHS, COMPLEX*16 A, INTEGER LDA, INTEGER LH, ↵
    INTEGER IPIVH, COMPLEX*16 B, INTEGER LDB, INTEGER INFO)
```

142.4 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	int (IN) The number of right hand sides, i.e., the number of columns of the matrix B. $NRHS \geq 0$.
A	PLASMA_Complex64_t* (IN) The tile factors L and U from the factorization, computed by PLASMA_zgetrf.
LDA	int (IN) The leading dimension of the array A. $LDA \geq \max(1, N)$.
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.
LDB	int (IN) The leading dimension of the array B. $LDB \geq \max(1, N)$.

142.5 Return Value:

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

142.6 Online Browsing

Dive into [PLASMA_zgets](#)

143 PLASMA_zgetsr Tile

143.1 Purpose

PLASMA_zgetsr 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.

143.2 C Bindings

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

143.3 Fortran Bindings

```
PLASMA_ZGETRS_TILE(INTEGER*4 A, INTEGER*4 L, INTEGER IPIVH, INTEGER*4 B, INTEGER INFO)
```

143.4 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.

143.5 Return Value:

```
= 0: successful exit
```

143.6 Online Browsing

Dive into [PLASMA_zgetrs_Tile](#)

144 PLASMA_zposv

144.1 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 $A * X = B$.

144.2 C Bindings

```
int PLASMA_zposv(PLASMA_enum uplo, int N, int NRHS, PLASMA_Complex64_t *A, int LDA,  $\leftrightarrow$ 
    PLASMA_Complex64_t *B, int LDB)
```

144.3 Fortran Bindings

```
PLASMA_ZPOSV(INTEGER uplo, INTEGER N, INTEGER NRHS, COMPLEX*16 A, INTEGER LDA, COMPLEX*16 B ↵
, INTEGER LDB, INTEGER INFO)
```

144.4 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	<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 ↵ ≥ 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	<code>int</code> (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	<code>int</code> (IN) The leading dimension of the array B. $LDB \geq \max(1, N)$.

144.5 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 ↵
.
```

144.6 Online Browsing

Dive into [PLASMA_zposv](#)

145 PLASMA_zposv_Tile

145.1 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.

145.2 C Bindings

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

145.3 Fortran Bindings

```
PLASMA_ZPOSV_TILE(INTEGER uplo, INTEGER*4 A, INTEGER*4 B, INTEGER INFO)
```

145.4 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 .

145.5 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 ↔
.
```

145.6 Online Browsing

Dive into [PLASMA_zposv_Tile](#)

146 PLASMA_zpotrf

146.1 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

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

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

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

146.2 C Bindings

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

146.3 Fortran Bindings

```
PLASMA_ZPOTRF(INTEGER uplo, INTEGER N, COMPLEX*16 A, INTEGER LDA, INTEGER INFO)
```

146.4 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. $LDA \geq \max(1, N)$.

146.5 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 ←
      .

```

146.6 Online Browsing

Dive into [PLASMA_zpotrf](#)

147 PLASMA_zpotrf_Tile

147.1 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.

147.2 C Bindings

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

147.3 Fortran Bindings

```
PLASMA_ZPOTRF_TILE(INTEGER uplo, INTEGER*4 A, INTEGER INFO)
```

147.4 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.

147.5 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 ↩
      .

```

147.6 Online Browsing

Dive into [PLASMA_zpotrf_Tile](#)

148 PLASMA_zpotrs

148.1 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**.

148.2 C Bindings

```

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

```

148.3 Fortran Bindings

```

PLASMA_ZPOTRS(INTEGER uplo, INTEGER N, INTEGER NRHS, COMPLEX*16 A, INTEGER LDA, COMPLEX*16 ↩
              B, INTEGER LDB, INTEGER INFO)

```

148.4 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	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 .
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 >= max(1,N).
```

148.5 Return Value:

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

148.6 Online Browsing

Dive into [PLASMA_zpotrs](#)

149 PLASMA_zpotrs_Tile

149.1 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.

149.2 C Bindings

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

149.3 Fortran Bindings

```
PLASMA_ZPOTRS_TILE(INTEGER uplo, INTEGER*4 A, INTEGER*4 B, INTEGER INFO)
```

149.4 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 $A = L*L \leftrightarrow **H$, 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.

149.5 Return Value:

```
= 0: successful exit
```

149.6 Online Browsing

Dive into [PLASMA_zpotrs_Tile](#)

150 PLASMA_ztrsm

150.1 Purpose

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

150.2 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)
```

150.3 Fortran Bindings

```
PLASMA_ZTRSM(INTEGER side, INTEGER uplo, INTEGER transA, INTEGER diag, INTEGER N, INTEGER ↵
    NRHS, COMPLEX*16 A, INTEGER LDA, COMPLEX*16 B, INTEGER LDB, INTEGER INFO)
```

150.4 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 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 ↵ ≥ 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 >= 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 >= max(1,N)`.

150.5 Return Value:

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

150.6 Online Browsing

Dive into [PLASMA_ztrsm](#)

151 PLASMA_ztrsm_Tile

151.1 Purpose

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

151.2 C Bindings

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

151.3 Fortran Bindings

```
PLASMA_ZTRSM_TILE(INTEGER side, INTEGER uplo, INTEGER transA, INTEGER diag, INTEGER*4 A, ←  
INTEGER*4 B, INTEGER INFO)
```

151.4 Arguments

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

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 is 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.

151.5 Return Value:

= 0: successful exit

151.6 Online Browsing

Dive into [PLASMA_ztrsm_Tile](#)

152 PLASMA_ztrsmpl

152.1 Purpose

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

152.2 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)
```

152.3 Fortran Bindings

```
PLASMA_ZTRSMPL(INTEGER N, INTEGER NRHS, COMPLEX*16 A, INTEGER LDA, INTEGER LH, INTEGER ↩
IPIVH, COMPLEX*16 B, INTEGER LDB, INTEGER INFO)
```

152.4 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>PLASMA_Complex64_t*</code> (IN) The tile factor L 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 right hand side matrix B. On exit, if <code>return</code> value = 0, the N-by-NRHS solution matrix X.
LDB	<code>PLASMA_Complex64_t*</code> (IN) The leading dimension of the array B. $LDB \geq \max(1, N)$.

152.5 Return Value:

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

152.6 Online Browsing

Dive into [PLASMA_ztrsmpi](#)

153 PLASMA_ztrsmpi_Tile

153.1 Purpose

PLASMA_ztrsmpi_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.

153.2 C Bindings

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

153.3 Fortran Bindings

```
PLASMA_ZTRSMPL_TILE(INTEGER*4 A, INTEGER*4 L, INTEGER IPIV, INTEGER*4 B, INTEGER INFO)
```

153.4 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.

153.5 Return Value:

= 0: successful exit

153.6 Online Browsing

Dive into [PLASMA_ztrsmpi_Tile](#)

154 PLASMA_zunglq

154.1 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.

154.2 C Bindings

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

154.3 Fortran Bindings

```
PLASMA_ZUNGLQ(INTEGER M, INTEGER N, INTEGER K, COMPLEX*16 A, INTEGER LDA, INTEGER T,  $\leftrightarrow$ 
COMPLEX*16 B, INTEGER LDB, INTEGER INFO)
```

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

154.5 Return Value:

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

154.6 Online Browsing

Dive into [PLASMA_zunglq](#)

155 PLASMA_zunglq_Tile

155.1 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.

155.2 C Bindings

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

155.3 Fortran Bindings

155.4 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.

155.5 Return Value:

= 0: successful exit

155.6 Online Browsing

Dive into [PLASMA_zunglq_Tile](#)

156 PLASMA_zungqr

156.1 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`.

156.2 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)
```

156.3 Fortran Bindings

```
PLASMA_ZUNGQR(INTEGER M, INTEGER N, INTEGER K, COMPLEX*16 A, INTEGER LDA, INTEGER T, ←
COMPLEX*16 B, INTEGER LDB, INTEGER INFO)
```

156.4 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 <code>matrix Q</code> . $M \geq K \geq 0$.

A	PLASMA_Complex64_t* (IN) Details of the QR factorization of the original matrix A as returned by <code>PLASMA_zgeqrf</code> .
LDA	<code>int</code> (IN) The leading dimension of the array A. <code>LDA >= max(1,M)</code> .
T	PLASMA_Complex64_t* (IN) Auxiliary factorization data, computed by <code>PLASMA_zgeqrf</code> .
B	PLASMA_Complex64_t* (OUT) On exit, the M-by-N matrix Q.
LDB	<code>int</code> (IN) The leading dimension of the array B. <code>LDB >= max(1,M)</code> .

156.5 Return Value:

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

156.6 Online Browsing

Dive into [PLASMA_zungqr](#)

157 PLASMA_zungqr_Tile

157.1 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.

157.2 C Bindings

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

157.3 Fortran Bindings

```
PLASMA_ZUNGQR_TILE(INTEGER*4 A, INTEGER*4 T, INTEGER*4 B, INTEGER INFO)
```

157.4 Arguments

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

157.5 Return Value:

= 0: successful exit

157.6 Online Browsing

Dive into [PLASMA_zungqr_Tile](#)

158 PLASMA_zunmlq

158.1 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.

158.2 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)
```

158.3 Fortran Bindings

```
PLASMA_ZUNMLQ(INTEGER side, INTEGER trans, INTEGER M, INTEGER N, INTEGER K, COMPLEX*16 A, ↵
    INTEGER LDA, INTEGER T, COMPLEX*16 B, INTEGER LDB, INTEGER INFO)
```

158.4 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 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	<code>int</code> (IN) The leading dimension of the array A. $LDA \geq \max(1, K)$.
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 .
LDB	<code>int</code> (IN) The leading dimension of the array C. $LDB \geq \max(1, M)$.

158.5 Return Value:

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

158.6 Online Browsing

Dive into [PLASMA_zunmlq](#)

159 PLASMA_zunmlq_Tile

159.1 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.

159.2 C Bindings

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

159.3 Fortran Bindings

```
PLASMA_ZUNMLQ_TILE(INTEGER side, INTEGER trans, INTEGER*4 A, INTEGER*4 T, INTEGER*4 B, INTEGER INFO)
```

159.4 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 LQ factorization of the original matrix A as returned by <code>↩</code> <code>PLASMA_zgelqf</code> .
T	PLASMA_Complex64_t* (IN) Auxiliary factorization data, computed by <code>PLASMA_zgelqf</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.

159.5 Return Value:

= 0: successful exit

159.6 Online Browsing

Dive into [PLASMA_zunmlq_Tile](#)

160 PLASMA_zunmqr

160.1 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.

160.2 C Bindings

```
int PLASMA_zunmqr(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)
```

160.3 Fortran Bindings

```
PLASMA_ZUNMQR(INTEGER side, INTEGER trans, INTEGER M, INTEGER N, INTEGER K, COMPLEX*16 A, ↩
    INTEGER LDA, INTEGER T, COMPLEX*16 B, INTEGER LDB, INTEGER INFO)
```

160.4 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 . $LDB \geq \max(1, M)$.

160.5 Return Value:

```

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

```

160.6 Online Browsing

Dive into [PLASMA_zunmqf](#)

161 PLASMA_zunmqr_Tile

161.1 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.

161.2 C Bindings

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

161.3 Fortran Bindings

```
PLASMA_ZUNMQR_TILE(INTEGER side, INTEGER trans, INTEGER*4 A, INTEGER*4 T, INTEGER*4 B, ↵
    INTEGER INFO)
```

161.4 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 ↵ PLASMA_zgeqrf.
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 $Q^{*H}B$.

161.5 Return Value:

= 0: successful exit

161.6 Online Browsing

Dive into [PLASMA_zunmqr_Tile](#)