testing_sgetrf_f.f90

```
!
!   -- MAGMA (version 1.3.0) --
!      Univ. of Tennessee, Knoxville
!      Univ. of California, Berkeley
!      Univ. of Colorado, Denver
!      November 2012
!
!  @generated s Wed Nov 14 22:54:27 2012
!
      program testing_sgetrf_f

      use magma

      external slamch, slange, sgemm, sgesv, sgetrs

      real slange, slamch

      real              :: rnumber(2), Anorm, Bnorm, Xnorm, Rnorm
      real, allocatable :: work(:)
      real, allocatable       :: A(:), B(:), X(:)
      real, allocatable       :: A2(:)
      integer,    allocatable       :: ipiv(:)

      real                    :: zone, mzone
      integer                       :: i, n, info, lda
      integer                       :: nrhs
      real(kind=8)                  :: flops, t, tstart, tend

      PARAMETER          ( nrhs = 1, zone = 1., mzone = -1. )

      call cublas_init()

      n   = 2048
      lda = n

!------ Allocate CPU memory
      allocate(A(lda*n))
      allocate(A2(lda*n))
      allocate(B(lda*nrhs))
      allocate(X(lda*nrhs))
      allocate(ipiv(n))
      allocate(work(n))

!---- Initializa the matrix
      do i=1,n*n
         call random_number(rnumber)
         A(i) = rnumber(1)
      end do
      A2(:) = A(:)

      do i=1,n*nrhs
         call random_number(rnumber)
         B(i) = rnumber(1)
      end do
      X(:) = B(:)

!---- Call magma LU ----------------
      call magma_wtime_f(tstart)
      call magmaf_sgetrf(n, n, A, lda, ipiv, info)
      call magma_wtime_f(tend)

      if ( info .ne. 0 )  then
         write(*,*) "Info : ", info
      end if

!---- Call solve -------------
      call sgetrs('n', n, nrhs, A, lda, ipiv, X, lda, info)
```

```
68
69          if ( info .ne. 0 )  then
70             write(*,*) "Info : ", info
71          end if
72
73    !---- Compare the two results ------
74          Anorm = slange('I', n, n,    A2, lda, work)
75          Bnorm = slange('I', n, nrhs, B,  lda, work)
76          Xnorm = slange('I', n, nrhs, X,  lda, work)
77
78          call sgemm('n', 'n', n,  nrhs, n, zone, A2, lda, X, lda, mzone, B, lda)
79          Rnorm = slange('I', n, nrhs, B, lda, work)
80
81          write(*,*)
82          write(*,*  ) 'Solving A x = b using LU factorization:'
83          write(*,105) '  || A || = ', Anorm
84          write(*,105) '  || b || = ', Bnorm
85          write(*,105) '  || x || = ', Xnorm
86          write(*,105) '  || b - A x || = ', Rnorm
87
88          flops = 2. * n * n * n / 3.
89          t = tend - tstart
90
91          write(*,*)   '  Gflops  = ',  flops / t / 1e9
92          write(*,*)
93
94          Rnorm = Rnorm / ( (Anorm*Xnorm+Bnorm) * n * slamch('E') )
95
96          if ( Rnorm > 60. ) then
97             write(*,105) '  Solution is suspicious, ', Rnorm
98          else
99             write(*,105) '  Solution is CORRECT'
100         end if
101
102   !---- Free CPU memory
103         deallocate(A, A2, B, X, ipiv, work)
104
105   !---- Free GPU memory
106         call cublas_shutdown()
107
108    105  format((a35,es10.3))
109
110         end
111
```

testing_zgetrf_gpu_f.f90

```
!
!   -- MAGMA (version 1.3.0) --
!      Univ. of Tennessee, Knoxville
!      Univ. of California, Berkeley
!      Univ. of Colorado, Denver
!      November 2012
!
!  @precisions normal z -> c d s
!
      program testing_zgetrf_gpu_f

      use magma

      external cublas_init, cublas_set_matrix, cublas_get_matrix
      external cublas_shutdown, cublas_alloc
      external zlange, zgemm, zgesv, dlamch

      double precision zlange, dlamch
      integer cublas_alloc

      double precision              :: rnumber(2), Anorm, Bnorm, Rnorm, Xnorm
      double precision, allocatable :: work(:)
      complex*16, allocatable       :: h_A(:), h_B(:), h_X(:)
      magma_devptr_t                :: devptrA, devptrB
      integer,    allocatable       :: ipiv(:)

      complex*16                    :: zone, mzone
      integer                       :: i, n, info, stat, lda
      integer                       :: size_of_elt, nrhs
      real(kind=8)                  :: flops, t, tstart, tend

      PARAMETER          ( nrhs = 1, zone = 1., mzone = -1. )

      call cublas_init()

      n   = 2048
      lda  = n
      ldda = ((n+31)/32)*32
      size_of_elt = sizeof_complex_16

!------ Allocate CPU memory
      allocate(h_A(lda*n))
      allocate(h_B(lda*nrhs))
      allocate(h_X(lda*nrhs))
      allocate(work(n))
      allocate(ipiv(n))

!------ Allocate GPU memory
      stat = cublas_alloc(ldda*n, size_of_elt, devPtrA)
      if (stat .ne. 0) then
         write(*,*) "device memory allocation failed"
         stop
      endif

      stat = cublas_alloc(ldda*nrhs, size_of_elt, devPtrB)
      if (stat .ne. 0) then
         write(*,*) "device memory allocation failed"
         stop
      endif

!---- Initializa the matrix
      do i=1,lda*n
         call random_number(rnumber)
         h_A(i) = rnumber(1)
      end do

      do i=1,lda*nrhs
```

```
68            call random_number(rnumber)
69            h_B(i) = rnumber(1)
70          end do
71          h_X(:) = h_B(:)
72
73    !---- devPtrA = h_A
74          call cublas_set_matrix(n, n, size_of_elt, h_A, lda, devptrA, ldda)
75
76    !---- devPtrB = h_B
77          call cublas_set_matrix(n, nrhs, size_of_elt, h_B, lda, devptrB, ldda)
78
79    !---- Call magma LU ----------------
80          call magma_wtime_f(tstart)
81          call magmaf_zgetrf_gpu(n, n, devptrA, ldda, ipiv, info)
82          call magma_wtime_f(tend)
83
84          if ( info .ne. 0 )  then
85             write(*,*) "Info : ", info
86          end if
87
88    !---- Call magma solve -------------
89          call magmaf_zgetrs_gpu('n', n, nrhs, devptrA, ldda, ipiv, devptrB, ldda, info)
90
91          if ( info .ne. 0 )  then
92             write(*,*) "Info : ", info
93          end if
94
95    !---- h_X = devptrB
96          call cublas_get_matrix (n, nrhs, size_of_elt, devptrB, ldda, h_X, lda)
97
98    !---- Compare the two results ------
99          Anorm = zlange('I', n, n,    h_A, lda, work)
100         Bnorm = zlange('I', n, nrhs, h_B, lda, work)
101         Xnorm = zlange('I', n, nrhs, h_X, lda, work)
102         call zgemm('n', 'n', n,  nrhs, n, zone, h_A, lda, h_X, lda, mzone, h_B, lda)
103         Rnorm = zlange('I', n, nrhs, h_B, lda, work)
104
105         write(*,*)
106         write(*,*  ) 'Solving A x = b using LU factorization:'
107         write(*,105) '  || A || = ', Anorm
108         write(*,105) '  || b || = ', Bnorm
109         write(*,105) '  || x || = ', Xnorm
110         write(*,105) '  || b - A x || = ', Rnorm
111
112         flops = 2. * n * n * n / 3.
113         t = tend - tstart
114
115         write(*,*)   '  Gflops  = ',  flops / t / 1e9
116         write(*,*)
117
118         Rnorm = Rnorm / ( (Anorm*Xnorm+Bnorm) * n * dlamch('E') )
119
120         if ( Rnorm > 60. ) then
121            write(*,105) '  Solution is suspicious, ', Rnorm
122         else
123            write(*,105) '  Solution is CORRECT'
124         end if
125
126   !---- Free CPU memory
127         deallocate(h_A, h_X, h_B, work, ipiv)
128
129   !---- Free GPU memory
130         call cublas_free(devPtrA)
131         call cublas_free(devPtrB)
132         call cublas_shutdown()
133
```

```
105  format((a35,es10.3))

     end

```

example_dgesv_f.f

```
1         PROGRAM EXAMPLE_DGESV_F
2   *
3   ************************************************************************
4   *     PLASMA example routine (version 2.5.0)
5   *     Author: Bilel Hadri
6   *     Release Date: November, 15th 2010
7   *     PLASMA is a software package provided by Univ. of Tennessee,
8   *     Univ. of California Berkeley and Univ. of Colorado Denver.
9   *     @generated d Thu Nov  8 11:44:47 2012
10  ************************************************************************
11  *
12        IMPLICIT NONE
13  *
14        INCLUDE "plasmaf.h"
15  *
16  *  Purpose
17  *  =======
18  *
19  *   FORTRAN EXAMPLE FOR PLASMA_DGESV
20  *   Example for solving a system of linear equations using LU
21  *
22  *  ========================================================================
23  *
24  *     .. Parameters ..
25        INTEGER           CORES, N, NRHS
26        PARAMETER         ( CORES = 2 )
27        PARAMETER         ( N = 10 )
28        PARAMETER         ( NRHS = 5 )
29        COMPLEX*16        ZONE
30        PARAMETER         ( ZONE = ( 1.0D+0, 0.0D+0 ) )
31  *     ..
32  *     .. Local Scalars ..
33        COMPLEX*16        A1( N, N ), B1( N, NRHS )
34        COMPLEX*16        A2( N, N ), B2( N, NRHS )
35        DOUBLE PRECISION  RWORK( N )
36        INTEGER*4         HL( 2 ), HPIV( 2 )
37        INTEGER           I, INFO
38        INTEGER           ISEED( 4 )
39        DOUBLE PRECISION  XNORM, ANORM, BNORM, RNORM, EPS
40        DOUBLE PRECISION  DLAMCH, DLANGE
41  *     ..
42  *     .. External Subroutines ..
43        EXTERNAL          ZLARNV, DLAMCH, DLANGE
44        EXTERNAL          PLASMA_INIT, PLASMA_ALLOC_WORKSPACE_DGESV_INCPIV
45        EXTERNAL          PLASMA_DGESV, PLASMA_FINALIZE
46        EXTERNAL          PLASMA_DEALLOC_HANDLE
47        EXTERNAL          DGEMM
48  *     ..
49  *     .. Executable Statements ..
50  *
51        DO  I = 1, 4
52            ISEED( I ) = 1
53        ENDDO
54  *
55  *     Initialize Plasma
56  *
57        CALL PLASMA_INIT( CORES, INFO )
58        WRITE(*,*) "-- PLASMA is initialized on", CORES, "cores."
59  *
60  *     Initialization of the matrix A1
61  *
62        CALL ZLARNV( 1, ISEED, N*N, A1 )
63        A2(:,:)=A1(:,:)
64  *
65  *     Initialization of the RHS
66  *
67        CALL ZLARNV( 1, ISEED, N*NRHS, B1 )
```

```
68          B2(:,:)=B1(:,:)
69    *
70    *     Allocate L and IPIV
71    *
72          CALL PLASMA_ALLOC_WORKSPACE_DGESV_INCPIV( N, HL, HPIV, INFO )
73    *
74    *     PLASMA DGESV
75    *
76          CALL PLASMA_DGESV_INCPIV( N, NRHS, A2, N, HL, HPIV,B2, N, INFO )
77    *
78    *     Check the solution
79    *
80          XNORM = DLANGE('I',N, NRHS, B2, N, RWORK)
81          ANORM = DLANGE('I',N, N, A1, N, RWORK)
82          BNORM = DLANGE('I',N, NRHS, B1, N, RWORK)
83
84          CALL DGEMM('No transpose','No transpose', N, NRHS, N, ZONE,
85         $              A1, N, B2, N, -ZONE, B1, N)
86
87          RNORM = DLANGE('I',N, NRHS, B1, N, RWORK)
88
89          EPS= DLAMCH('Epsilon')
90
91          WRITE(*,*) '============'
92          WRITE(*,*) 'Checking the Residual of the solution '
93          WRITE(*,*) '-- ||Ax-B||_oo/((||A||_oo||x||_oo+||B||_oo).N.eps)=',
94         $        RNORM / ((ANORM * XNORM + BNORM) * N * EPS)
95
96          IF ((RNORM > 60.0).AND.( INFO < 0 )) THEN
97              WRITE(*,*) "-- Error in DGESV example !"
98          ELSE
99              WRITE(*,*) "-- Run of DGESV example successful !"
100         ENDIF
101
102   *
103   *     Deallocate L and IPIV
104   *
105         CALL PLASMA_DEALLOC_HANDLE( HL, INFO )
106         CALL PLASMA_DEALLOC_HANDLE( HPIV, INFO )
107   *
108   *     Finalize Plasma
109   *
110         CALL PLASMA_FINALIZE( INFO )
111   *
112   *     End of EXAMPLE_DGESV.
113   *
114         END PROGRAM EXAMPLE_DGESV_F
115
```