CPS 5401 Qualifying Exam Name ____________________
Shirley Moore, Instructor
May 26, 2015 10:00am-12:45pm

Choose 3 out of the 4 questions to answer.

1. Consider a multicore processor with 8 cores, with each core having a clock speed of 2.6 GHz and 256-bit wide vectorization capability. The measured maximum memory bandwidth is 56 GB/sec.

a. Sketch the roofline model for the processor and identify the machine balance point.

b. Write code in the language of your choice for the 2-loop formulation of matrix-vector multiply. Find the operational intensity of the code (i.e., the ratio of floating point operations to amount of data moved) and determine the maximum performance given by the roofline model.

c. Assuming that three 50x50 double-precision matrices fit in fast cache memory, find the operational intensity of a cache-blocked matrix-matrix multiply code and determine the maximum performance given by the roofline model.

2. Four variations of an OpenMP code to calculate the number of hits for Monte Carlo computation of π are shown below.

a. Which of the codes will produce a correct result? If a variation is incorrect, explain why.

b. Which of the codes would you expect to have the best performance? If a variation is correct but is expected to have less than optimal performance, explain why (for each such variation).

Code 1:
```
#pragma omp parallel for private (i, x, y, seed) shared (ntrials, hits)
for (i = 0; i < ntrials; i++) {
   x = ((double) rand_r(&seed))/ RAND_MAX;
   y = ((double) rand_r(&seed))/ RAND_MAX;
   if ((x*x + y*y) <= 1.0)
      hits++;
}
```

Code 2:
```
#pragma omp parallel for private (i, x, y, seed) shared (ntrials, hits)
for (i = 0; i < ntrials; i++) {
   x = ((double) rand_r(&seed))/ RAND_MAX;
   y = ((double) rand_r(&seed))/ RAND_MAX;
   if ((x*x + y*y) <= 1.0)
      #pragma omp critical
      hits++;
}
```

Code 3:
```
#pragma omp parallel for private (i, x, y, seed) shared (ntrials, myhits)
for (i = 0; i < ntrials; i++) {
    x = ((double) rand_r(&seed))/ RAND_MAX;
    y = ((double) rand_r(&seed))/ RAND_MAX;
    if ((x*x + y*y) <= 1.0)
        hits[i]++;
}
hits = 0;
for (i = 0; i < nthreads; i++);
   hits += myhits[i];
```

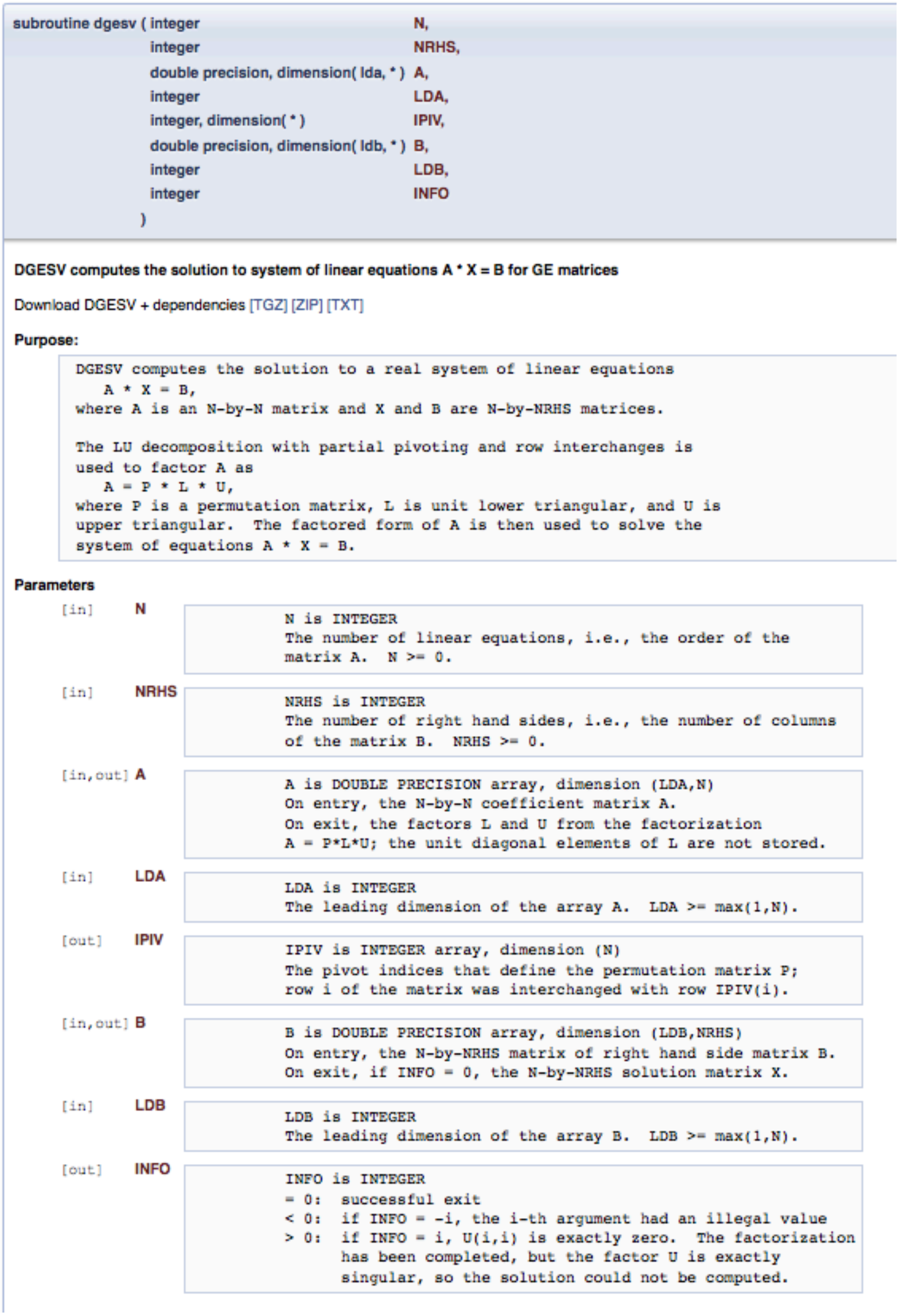
Code 4:
```
#pragma omp parallel for private(i, x, y, seed) shared(ntrials) reduction(+: hits)
for (i = 0; i < ntrials; i++) {
    x = ((double) rand_r(&seed))/ RAND_MAX;
    y = ((double) rand_r(&seed))/ RAND_MAX;
    if ((x*x + y*y) <= 1.0)
        hits++;
}
```

3. For each of the MPI codes below, explain whether it is safe or not. Safety means that the code is free from possibility of deadlock and that buffer contents still in use are not overwritten. Assume that numtasks hold the number of MPI tasks and taskid holds the rank of the processor executing the code. If the code is incorrect, rewrite it so that it is correct.

a.
```
MPI_Send(&buf, count, MPI_DOUBLE, (i+1)%numtasks, 1, MPI_COMM_WORLD);
MPI_Recv(&buf, count, MPI_DOUBLE, (i-1+numtasks)%numtasks, 1, MPI_COMM_WORLD);
```

b.
```
 MPI_Isend(&buf, count, MPI_DOUBLE, (i+1)%numtasks, 1, MPI_COMM_WORLD, &request);
 MPI_Recv(&buf, count, MPI_DOUBLE, (i-1+numtasks)%numtasks, 1, MPI_COMM_WORLD,
&status);
MPI_Wait(&request, &status);
```

4. Suppose you have a nonsingular square dense double precision real matrix A of size n x n and several right-hand side vectors b of size n for which you need to solve the system Ax=b. Choosing from the routines given, write code for how you would efficiently solve the set of linear systems. You need only show the code to solve one of the systems, but assume that several would be solved using the same matrix A.



```
subroutine dgesv ( integer                              N,
                   integer                              NRHS,
                   double precision, dimension( lda, * ) A,
                   integer                              LDA,
                   integer, dimension( * )              IPIV,
                   double precision, dimension( ldb, * ) B,
                   integer                              LDB,
                   integer                              INFO
                 )
```

DGESV computes the solution to system of linear equations A * X = B for GE matrices

Download DGESV + dependencies [TGZ] [ZIP] [TXT]

Purpose:

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

Parameters

[in] **N**

```
N is INTEGER
The number of linear equations, i.e., the order of the
matrix A.  N >= 0.
```

[in] **NRHS**

```
NRHS is INTEGER
The number of right hand sides, i.e., the number of columns
of the matrix B.  NRHS >= 0.
```

[in,out] **A**

```
A is DOUBLE PRECISION array, dimension (LDA,N)
On entry, the N-by-N coefficient matrix A.
On exit, the factors L and U from the factorization
A = P*L*U; the unit diagonal elements of L are not stored.
```

[in] **LDA**

```
LDA is INTEGER
The leading dimension of the array A.  LDA >= max(1,N).
```

[out] **IPIV**

```
IPIV is INTEGER array, dimension (N)
The pivot indices that define the permutation matrix P;
row i of the matrix was interchanged with row IPIV(i).
```

[in,out] **B**

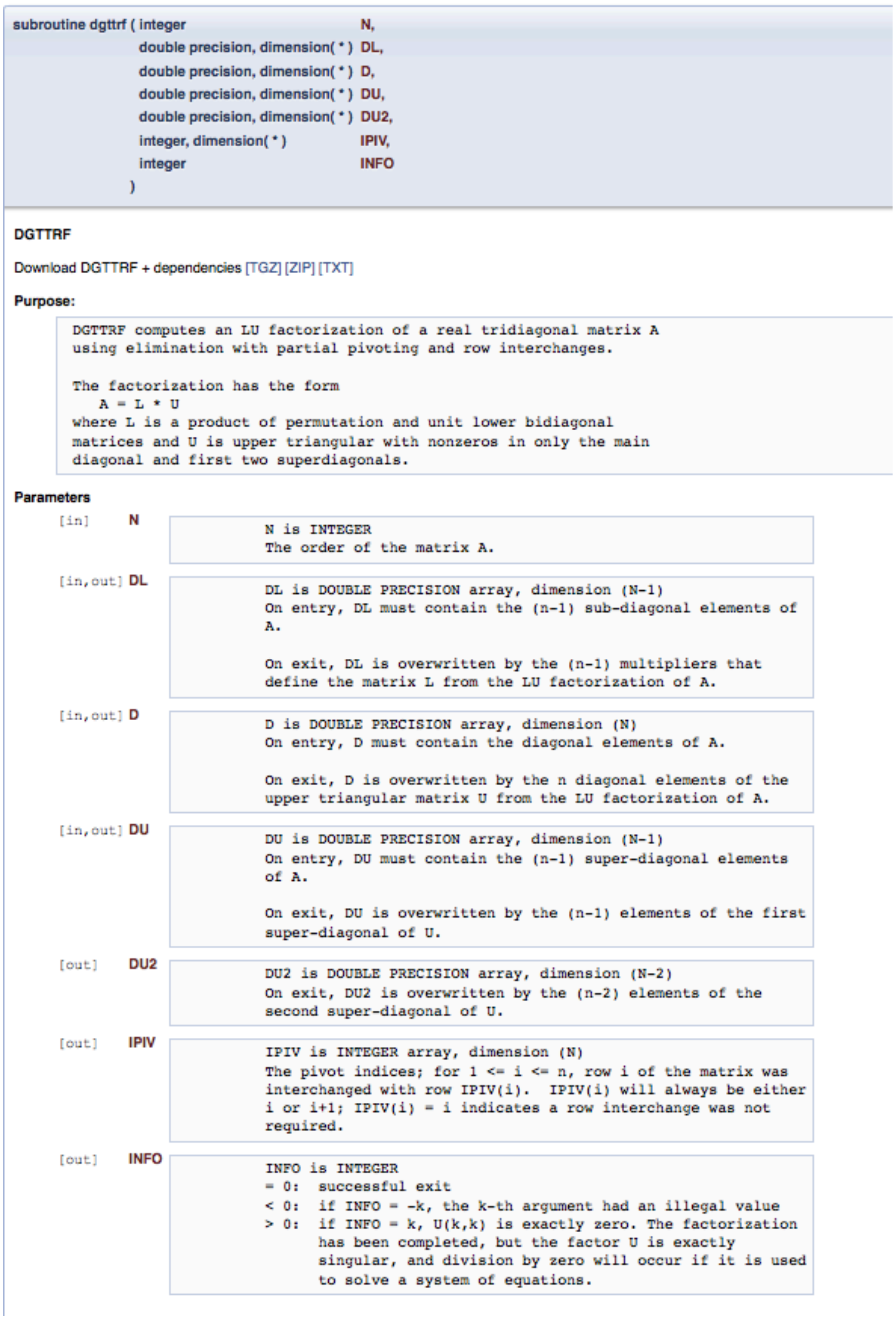
```
B is DOUBLE PRECISION array, dimension (LDB,NRHS)
On entry, the N-by-NRHS matrix of right hand side matrix B.
On exit, if INFO = 0, the N-by-NRHS solution matrix X.
```

[in] **LDB**

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

[out] **INFO**

```
INFO is INTEGER
= 0:  successful exit
< 0:  if INFO = -i, the i-th argument had an illegal value
> 0:  if INFO = 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.
```



```
subroutine dgttrf ( integer                            N,
                    double precision, dimension( * )  DL,
                    double precision, dimension( * )  D,
                    double precision, dimension( * )  DU,
                    double precision, dimension( * )  DU2,
                    integer, dimension( * )            IPIV,
                    integer                            INFO
                  )
```

DGTTRF

Download DGTTRF + dependencies [TGZ] [ZIP] [TXT]

Purpose:

```
DGTTRF computes an LU factorization of a real tridiagonal matrix A
using elimination with partial pivoting and row interchanges.

The factorization has the form
   A = L * U
where L is a product of permutation and unit lower bidiagonal
matrices and U is upper triangular with nonzeros in only the main
diagonal and first two superdiagonals.
```

Parameters

[in] **N**

```
N is INTEGER
The order of the matrix A.
```

[in,out] **DL**

```
DL is DOUBLE PRECISION array, dimension (N-1)
On entry, DL must contain the (n-1) sub-diagonal elements of
A.

On exit, DL is overwritten by the (n-1) multipliers that
define the matrix L from the LU factorization of A.
```

[in,out] **D**

```
D is DOUBLE PRECISION array, dimension (N)
On entry, D must contain the diagonal elements of A.

On exit, D is overwritten by the n diagonal elements of the
upper triangular matrix U from the LU factorization of A.
```

[in,out] **DU**

```
DU is DOUBLE PRECISION array, dimension (N-1)
On entry, DU must contain the (n-1) super-diagonal elements
of A.

On exit, DU is overwritten by the (n-1) elements of the first
super-diagonal of U.
```

[out] **DU2**

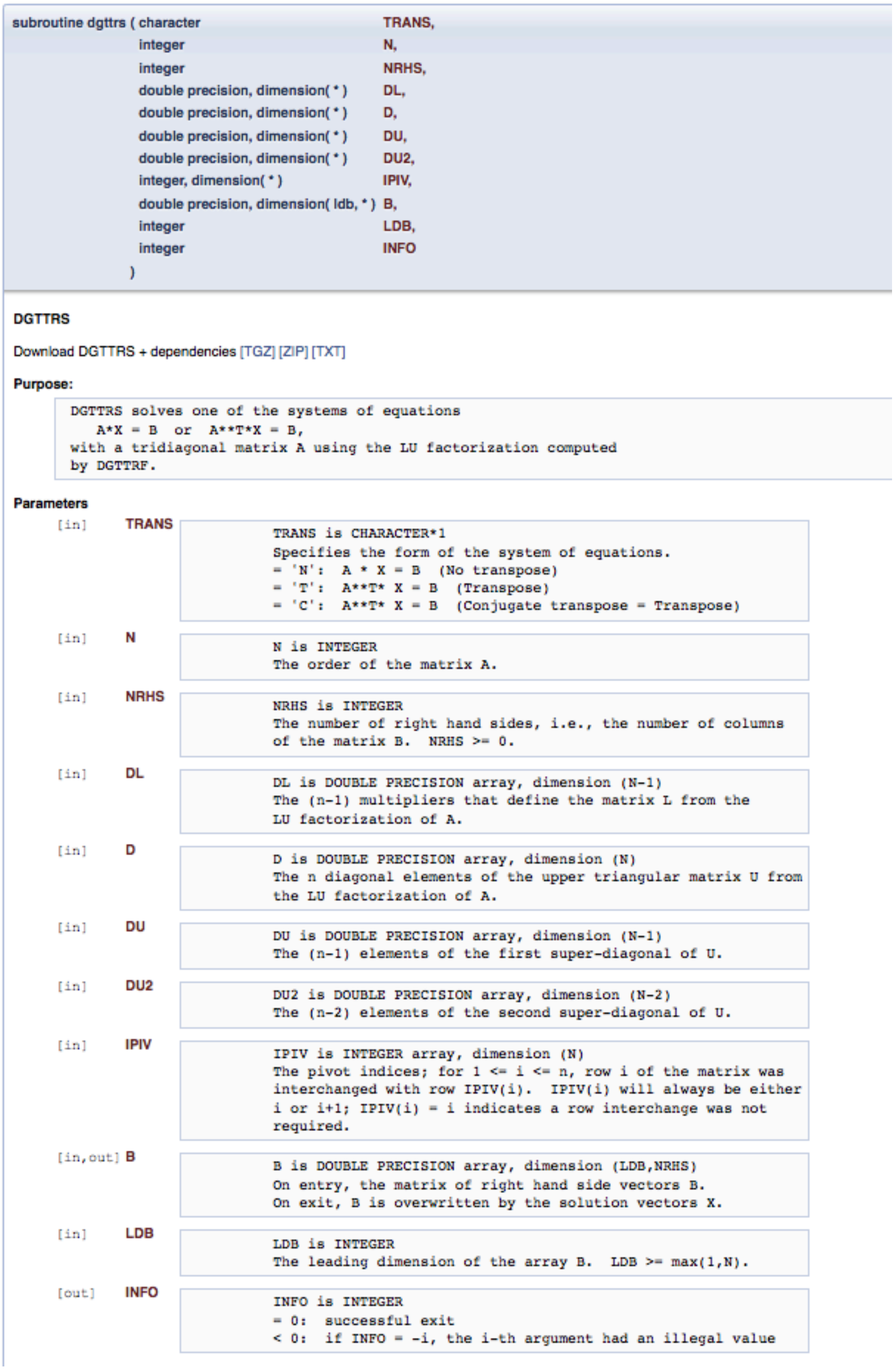
```
DU2 is DOUBLE PRECISION array, dimension (N-2)
On exit, DU2 is overwritten by the (n-2) elements of the
second super-diagonal of U.
```

[out] **IPIV**

```
IPIV is INTEGER array, dimension (N)
The pivot indices; for 1 <= i <= n, row i of the matrix was
interchanged with row IPIV(i).  IPIV(i) will always be either
i or i+1; IPIV(i) = i indicates a row interchange was not
required.
```

[out] **INFO**

```
INFO is INTEGER
= 0:  successful exit
< 0:  if INFO = -k, the k-th argument had an illegal value
> 0:  if INFO = k, U(k,k) 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.
```



```
subroutine dgttrs ( character                              TRANS,
                    integer                                N,
                    integer                                NRHS,
                    double precision, dimension( * )       DL,
                    double precision, dimension( * )       D,
                    double precision, dimension( * )       DU,
                    double precision, dimension( * )       DU2,
                    integer, dimension( * )                IPIV,
                    double precision, dimension( ldb, * )  B,
                    integer                                LDB,
                    integer                                INFO
                  )
```

DGTTRS

Download DGTTRS + dependencies [TGZ] [ZIP] [TXT]

Purpose:

```
DGTTRS solves one of the systems of equations
   A*X = B  or  A**T*X = B,
with a tridiagonal matrix A using the LU factorization computed
by DGTTRF.
```

Parameters

[in] **TRANS**

```
TRANS is CHARACTER*1
Specifies the form of the system of equations.
= 'N':  A * X = B  (No transpose)
= 'T':  A**T* X = B  (Transpose)
= 'C':  A**T* X = B  (Conjugate transpose = Transpose)
```

[in] **N**

```
N is INTEGER
The order of the matrix A.
```

[in] **NRHS**

```
NRHS is INTEGER
The number of right hand sides, i.e., the number of columns
of the matrix B.  NRHS >= 0.
```

[in] **DL**

```
DL is DOUBLE PRECISION array, dimension (N-1)
The (n-1) multipliers that define the matrix L from the
LU factorization of A.
```

[in] **D**

```
D is DOUBLE PRECISION array, dimension (N)
The n diagonal elements of the upper triangular matrix U from
the LU factorization of A.
```

[in] **DU**

```
DU is DOUBLE PRECISION array, dimension (N-1)
The (n-1) elements of the first super-diagonal of U.
```

[in] **DU2**

```
DU2 is DOUBLE PRECISION array, dimension (N-2)
The (n-2) elements of the second super-diagonal of U.
```

[in] **IPIV**

```
IPIV is INTEGER array, dimension (N)
The pivot indices; for 1 <= i <= n, row i of the matrix was
interchanged with row IPIV(i).  IPIV(i) will always be either
i or i+1; IPIV(i) = i indicates a row interchange was not
required.
```

[in,out] **B**

```
B is DOUBLE PRECISION array, dimension (LDB,NRHS)
On entry, the matrix of right hand side vectors B.
On exit, B is overwritten by the solution vectors X.
```

[in] **LDB**

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

[out] **INFO**

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