```
c===========================================================
c     Test program for LAPACK "driver" routine 'dgesv'
c     which computes the solution of a real system
c     of linear equations:  A x = b
c
c     This version uses fixed-size 2-d arrays.
c===========================================================
      program          tdgesv1

      implicit         none
c-----------------------------------------------------------
c     Maximum size of linear system.
c-----------------------------------------------------------
      integer          maxn
      parameter      ( maxn = 100 )

c-----------------------------------------------------------
c     Storage for arrays
c-----------------------------------------------------------
      real*8           a(maxn,maxn),
     &                 b(maxn),        x(maxn)
      integer          ipiv(maxn)

      integer          i,              nrhs,
     &                 n,              info
```

```
c-----------------------------------------------------------
c     Set up sample 3 x 3 system ...
c-----------------------------------------------------------
      a(1,1) =  1.23d0
      a(1,2) =  0.24d0
      a(1,3) = -0.45d0

      a(2,1) = -0.43d0
      a(2,2) =  2.45d0
      a(2,3) =  0.78d0

      a(3,1) =  0.51d0
      a(3,2) = -0.68d0
      a(3,3) =  3.23d0

      b(1)   =  6.78d0
      b(2)   = -3.45d0
      b(3)   =  1.67d0

c-----------------------------------------------------------
c     ... and solve it.  Note that 'dgsev' is general
c     enough to solve A x_i = b_i for multiple right-hand-
c     sides b_i.  Here we have only one right-hand-side.
c     Also note that the procedure performs the LU
c     decomposition in place, thus destroying the
c     input-matrix, it also overwrites the right-hand-side(s)
c     with the solution(s).  Finally, observe that we
c     pass the "leading dimension" (maxn) of both 'a' and
c     'b' to the routine.  This allows us to load array
c     elements in the main program as we have just done,
c     without running into troubles due to the fact that
c     these elements ARE NOT all contiguous in memory.
c-----------------------------------------------------------
```

```
      n    = 3
      nrhs = 1

      call dgesv( n, nrhs, a, maxn, ipiv, b, maxn, info )

      if(      info .eq. 0 ) then
c-----------------------------------------------------------
c        Solution successful, write soln to stdout.
c        Note the use of "implied-do-loop" to write a
c        sequence of elements: the enclosing parenthesis
c        around the  "loop" are required.
c-----------------------------------------------------------
         write(*,*) ( b(i) , i = 1 , n )
      else if( info .lt. 0 ) then
c-----------------------------------------------------------
c        Bad argument detected.
c-----------------------------------------------------------
         write(0,*) 'tdgesv1: Argument ', abs(info),
     &              ' to dgesv() is invalid'
      else
c-----------------------------------------------------------
c        Matrix is singular.
c-----------------------------------------------------------
         write(0,*) 'tdgesv1: dgesv() detected singular ',
     &              'matrix'
      end if


      stop

      end
```

```
      SUBROUTINE DGESV( N, NRHS, A, LDA, IPIV, B, LDB, INFO )
*
*  -- LAPACK driver routine (version 2.0) --
*     Univ. of Tennessee, Univ. of California Berkeley, NAG Ltd.,
*     Courant Institute, Argonne National Lab, and Rice University
*     March 31, 1993
*
*     .. Scalar Arguments ..
      INTEGER            INFO, LDA, LDB, N, NRHS
*     ..
*     .. Array Arguments ..
      INTEGER            IPIV( * )
      DOUBLE PRECISION   A( LDA, * ), B( LDB, * )
*     ..
*
*  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.
*
*  Arguments
*  =========
*
*  N       (input) INTEGER
*          The number of linear equations, i.e., the order of the
```

```
*          matrix A.  N >= 0.
*
*  NRHS    (input) INTEGER
*          The number of right hand sides, i.e., the number of columns
*          of the matrix B.  NRHS >= 0.
*
*  A       (input/output) 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.
*
*  LDA     (input) INTEGER
*          The leading dimension of the array A.  LDA >= max(1,N).
*
*  IPIV    (output) INTEGER array, dimension (N)
*          The pivot indices that define the permutation matrix P;
*          row i of the matrix was interchanged with row IPIV(i).
*
*  B       (input/output) 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.
*
*  LDB     (input) INTEGER
*          The leading dimension of the array B.  LDB >= max(1,N).
*
*  INFO    (output) 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.
*
*  =====================================================================
*
```

```
*     .. External Subroutines ..
      EXTERNAL           DGETRF, DGETRS, XERBLA
*     ..
*     .. Intrinsic Functions ..
      INTRINSIC          MAX
*     ..
*     .. Executable Statements ..
*
*     Test the input parameters.
*
      INFO = 0
      IF( N.LT.0 ) THEN
         INFO = -1
      ELSE IF( NRHS.LT.0 ) THEN
         INFO = -2
      ELSE IF( LDA.LT.MAX( 1, N ) ) THEN
         INFO = -4
      ELSE IF( LDB.LT.MAX( 1, N ) ) THEN
         INFO = -7
      END IF
      IF( INFO.NE.0 ) THEN
         CALL XERBLA( 'DGESV ', -INFO )
         RETURN
      END IF
*
*     Compute the LU factorization of A.
*
      CALL DGETRF( N, N, A, LDA, IPIV, INFO )
      IF( INFO.EQ.0 ) THEN
*
*        Solve the system A*X = B, overwriting B with X.
*
         CALL DGETRS( 'No transpose', N, NRHS, A, LDA, IPIV, B, LDB,
     $                INFO )
```

```
      END IF
      RETURN
*
*     End of DGESV
*
      END
```

```
#############################################################
# Building 'tdgesv' and sample output on the SGIs
#############################################################
einstein% pwd; ls
/usr2/people/phy329/linsys/ex1
Makefile     tdgesv.f

einstein% printenv LIBBLAS
-lblas

einstein% more Makefile
############################################################
#  Note that this 'Makefile' assumes that the following
#  environment variables are set:
#
#     F77
#     F77PP
#     F77FLAGS
#     F77CFLAGS
#     F77LFLAGS
#     LIBBLAS
#
#  Put the appropriate 'setenv' commands in your '~/.cshrc'.
#  See 'phy329@einstein:~/.cshrc' for an example.
#
#  IMPORTANT: Note the use of LIBBLAS which should be
#  set to '-lblas' on the SGI machines.  BLAS stands for
#  Basic Linear Algebra Support and is a Fortran- and
#  C-callable library which implements basic manipulations
#  useful in numerical linear algebra.
############################################################
```

```
.IGNORE:

F77_COMPILE  = $(F77) $(F77FLAGS) $(F77CFLAGS)
F77_LOAD     = $(F77) $(F77FLAGS) $(F77LFLAGS)

.f.o:
        $(F77PP) $*.f
        $(F77_COMPILE) $*.f

EXECUTABLES = tdgesv

all: $(EXECUTABLES)

tdgesv: tdgesv.o
        $(F77_LOAD) tdgesv.o -llapack $(LIBBLAS) -o tdgesv

clean:
        rm *.o
        rm $(EXECUTABLES)


##############################################################
# Build and run the test application ...
##############################################################
einstein% make
        touch tdgesv.f
        f77 -g -n32 -c tdgesv.f
        f77 -g -n32 -L/usr/localn32/lib -n32 tdgesv.o \
            -llapack -lblas -o tdgesv

einstein% tdgesv

    5.426364412431639     -0.3257753768173935     -0.4083508069894624
```

```
################################################################
# Building 'tdgesv' and sample output on the Cray J90.
################################################################
charon% pwd; ls
/archive/utexas/ph/phas761/phy329/linsys/ex1

################################################################
# Note that the 'blas' library is built-in on the Cray, so we
# do not have to specify '-lblas' during the load phase.
# Thus, the environment variable LIBBLAS (used in the Makefile)
# can either be unset, or set to nothing (the case here).
################################################################
charon% printenv LIBBLAS

################################################################
# Transfer the source code and the Makefile from einstein.
################################################################
charon% ftp einstein.ph.utexas.edu
Connected to einstein.ph.utexas.edu.
                            .
                            .
                            .
Remote system type is UNIX.
Using binary mode to transfer files.
Name (einstein.ph.utexas.edu:phas761): phy329
331 Password required for phy329.
Password:
230 User phy329 logged in.

ftp> cd linsys/ex1
250 CWD command successful.

ftp> prompt
Interactive mode off.
```

```
ftp> mget Makefile *.f
200 PORT command successful.
150 Opening BINARY mode data connection for 'Makefile' (977 bytes).
226 Transfer complete.
977 bytes received in 0.0024 seconds (3.9e+02 Kbytes/s)
200 PORT command successful.
150 Opening BINARY mode data connection for 'tdgesv.f' (3274 bytes).
226 Transfer complete.
3274 bytes received in 0.0017 seconds (1.9e+03 Kbytes/s)

ftp> quit
221 Goodbye.

#############################################################
# Using the 'Makefile', the source file is automatically
# translated # to 'Cray-compatible' Fortran via the script
# 'f77transcray'.  Note that 'f77transcray' handles the change
# in name of the real*8 LAPACK routine from 'dgesv' to 'sgesv'.
#############################################################
charon% make
   f77transcray tdgesv.f
f77transcray: Translating tdgesv.f
   cf77 -g -c tdgesv.f
   cf77 -g -L/home/utexas/ph/phas761/lib tdgesv.o -llapack  -o tdgesv
```

```
charon% cat tdgesv.f
c===========================================================
c     Test program for LAPACK "driver" routine 'dgesv'
c     which computes the solution of a real system
c     of linear equations:  A x = b
c
c     This version uses fixed-size 2-d arrays.
c===========================================================
      program         tdgesv1

      implicit        none
c-----------------------------------------------------------
c     Maximum size of linear system.
c-----------------------------------------------------------
      integer         maxn
      parameter     ( maxn = 100 )

c-----------------------------------------------------------
c     Storage for arrays
c-----------------------------------------------------------
      real            a(maxn,maxn),
     &                b(maxn),       x(maxn)
      integer         ipiv(maxn)

      integer         i,             nrhs,
     &                n,             info
```

```
c-----------------------------------------------------------
c     Set up sample 3 x 3 system ...
c-----------------------------------------------------------
      a(1,1) =  1.23e0
      a(1,2) =  0.24e0
      a(1,3) = -0.45e0

      a(2,1) = -0.43e0
      a(2,2) =  2.45e0
      a(2,3) =  0.78e0

      a(3,1) =  0.51e0
      a(3,2) = -0.68e0
      a(3,3) =  3.23e0

      b(1)   =  6.78e0
      b(2)   = -3.45e0
      b(3)   =  1.67e0

c-----------------------------------------------------------
c     ... and solve it.  Note that 'dgsev' is general
c     enough to solve A x_i = b_i for multiple right-hand-
c     sides b_i.  Here we have only one right-hand-side.
c     Also note that the procedure performs the LU
c     decomposition in place, thus destroying the
c     input-matrix, it also overwrites the right-hand-side(s)
c     with the solution(s).  Finally, observe that we
c     pass the "leading dimension" (maxn) of both 'a' and
c     'b' to the routine.  This allows us to load array
c     elements in the main program as we have just done,
c     without running into troubles due to the fact that
c     these elements ARE NOT all contiguous in memory.
c-----------------------------------------------------------
```

```
      n    = 3
      nrhs = 1

      call sgesv( n, nrhs, a, maxn, ipiv, b, maxn, info )

      if(      info .eq. 0 ) then
c-----------------------------------------------------------
c        Solution successful, write soln to stdout.
c        Note the use of "implied-do-loop" to write a
c        sequence of elements: the enclosing parenthesis
c        around the  "loop" are required.
c-----------------------------------------------------------
         write(*,*) ( b(i) , i = 1 , n )
      else if( info .lt. 0 ) then
c-----------------------------------------------------------
c        Bad argument detected.
c-----------------------------------------------------------
         write(0,*) 'tdgesv1: Argument ', abs(info),
     &              ' to sgesv() is invalid'
      else
c-----------------------------------------------------------
c        Matrix is singular.
c-----------------------------------------------------------
         write(0,*) 'tdgesv1: sgesv() detected singular ',
     &              'matrix'
      end if


      stop

      end
```

```
###############################################################
# Run the test program.
###############################################################
charon% tdgesv

 5.426364412431667,  -0.325775376817397,  -0.4083508069894641

 STOP executed at line 88 in Fortran routine 'TDGESV1'
 CPU: 0.006s,  Wallclock: 0.043s,  1.7% of 8-CPU Machine
 Memory HWM: 232358, Stack HWM: 16509, Stack segment expansions: 0
```