```
c===========================================================
c     Solves 1-d linear boundary value problem
c
c         u''(x) = f(x)  on  x = [0,1]; u(0) = u0, u(1) = u1
c
c     using second-order finite difference technique and
c     LAPACK tridiagonal solver DGTSV.
c===========================================================
      program         bvp1d

      implicit        none

      integer         i4arg
```

```
c-------------------------------------------------------------
c     Extrema of problem domain; note that this approach
c     of defining extrema as parameters makes it easier
c     to generalize program to arbitrary domains.
c-------------------------------------------------------------
      real*8          xmin,          xmax
      parameter     ( xmin = 0.0d0,  xmax = 1.0d0 )
c-------------------------------------------------------------
c     Maximum problem size (2**20 + 1)
c-------------------------------------------------------------
      integer         maxn
      parameter     ( maxn = 1 048 577 )
c-------------------------------------------------------------
c     Storage for discrete x-values, unknowns, exact
c     solution and right hand side values
c-------------------------------------------------------------
      real*8          x(maxn),       u(maxn),
     &                uexact(maxn),  f(maxn)
c-------------------------------------------------------------
c     Storage for main, upper and lower diagonals of
c     tridiagonal system, and right-hand-side vector
c     for use with LAPACK routine DGTSV
c-------------------------------------------------------------
      real*8          d(maxn),       du(maxn),
     &                dl(maxn),      rhs(maxn)
      integer         nrhs,          info
c-------------------------------------------------------------
c     Discretization level and size of system (# of discrete
c     unknowns)
c-------------------------------------------------------------
      integer         level,         n,             j,
     &                option
```

```
c-----------------------------------------------------------
c     Mesh spacing and related constants (1/h**2, -2/h**2)
c-----------------------------------------------------------
      real*8          h,              hm2,            m2hm2
      real*8          rmserr
c-----------------------------------------------------------
c     Argument parsing.
c-----------------------------------------------------------
      level = i4arg(1,-1)
      if( level .lt. 0 ) go to 900
      n = 2 ** level + 1
      if( n .gt. maxn ) then
         write(0,*) 'Insufficient internal storage'
         stop
      end if
      option = i4arg(2,0)
c-----------------------------------------------------------
c     Set up finite-difference 'mesh' (discrete x-values)
c     and define some useful constants.
c-----------------------------------------------------------
      h     =  1.0d0 / (n - 1)
      do j = 1 , n
         x(j) = xmin + (j - 1) * h
      end do
      hm2   =  1.0d0 / (h * h)
      m2hm2 = -2.0d0 / (h * h)
c-----------------------------------------------------------
c     This only ensures that x(n) = xmax EXACTLY and is not
c     essential.
c-----------------------------------------------------------
      x(n) = xmax
```

```
c-------------------------------------------------------------
c     Set up exact solution and right hand side vector.
c-------------------------------------------------------------
      call exact(uexact,f,x,n)

c=============================================================
c     Set up tridiagonal system.  Note that indexing on
c     lower diagonal is always (j-1) when implementing the
c     j'th equation.
c=============================================================

c-------------------------------------------------------------
c     Left boundary: u(1) = u_0
c-------------------------------------------------------------
      d(1)        = 1.0d0
      du(1)       = 0.0d0
      rhs(1)      = uexact(1)
c-------------------------------------------------------------
c     Interior: Second order FDA of ODE.
c-------------------------------------------------------------
      do j = 2 , n - 1
         dl(j-1) = hm2
         d(j)    = m2hm2
         du(j)   = hm2
         rhs(j)  = f(j)
      end do
c-------------------------------------------------------------
c     Right boundary: u(n) = u_1
c-------------------------------------------------------------
      dl(n-1)     = 0.0d0
      d(n)        = 1.0d0
      rhs(n)      = uexact(n)
```

```
c===========================================================
c     Solve tridiagonal system.
c===========================================================

      nrhs = 1
      call dgtsv( n, nrhs, dl, d, du, rhs, n, info )

      if( info .eq. 0 ) then
c-----------------------------------------------------------
c        Solver successful, output either (x_j, u_j) or
c        (x_j, error_j) to stdout.  Also compute rms error
c        and output to standard error.
c-----------------------------------------------------------
         rmserr = 0.0d0
         do j = 1 , n
            if( option .eq. 0 ) then
               write(*,*) x(j), rhs(j)
            else
               write(*,*) x(j), (uexact(j) - rhs(j))
            end if
            rmserr = rmserr + (uexact(j) - rhs(j)) ** 2
         end do
         rmserr = sqrt(rmserr / n)
         write(0,*) 'rmserr = ', rmserr
      else
c-----------------------------------------------------------
c        Solver failed.
c-----------------------------------------------------------
         write(0,*) 'bvp1d: dgtsv() failed, info = ', info
      end if

      stop
```

```
 900  continue
         write(0,*) 'usage: bvp1d <level> [<option>]'
         write(0,*)
         write(0,*) '       Specify option .ne. 0 for output'
         write(0,*) '       of error instead of solution'
      stop

      end

c===========================================================
c     Computes exact values for u(x) (unknown function)
c     and f(x) (right hand side function).  x array must
c     have been previously defined.
c===========================================================
      subroutine exact(u,f,x,n)

         implicit        none
         integer         n
         real*8          u(n),     f(n),    x(n)

         real*8          pi2
         integer         j

         pi2 = 8.0d0 * atan(1.0d0)
         do j = 1 , n
            u(j) = sin(pi2 * x(j))
            f(j) = -pi2 * pi2 * u(j)
         end do

         return

      end
```

```
      SUBROUTINE DGTSV( N, NRHS, DL, D, DU, B, LDB, INFO )
*
*  -- LAPACK routine (version 2.0) --
*     Univ. of Tennessee, Univ. of California Berkeley, NAG Ltd.,
*     Courant Institute, Argonne National Lab, and Rice University
*     September 30, 1994
*
*     .. Scalar Arguments ..
      INTEGER            INFO, LDB, N, NRHS
*     ..
*     .. Array Arguments ..
      DOUBLE PRECISION   B( LDB, * ), D( * ), DL( * ), DU( * )
*     ..
*
*  Purpose
*  =======
*
*  DGTSV  solves the equation
*
*     A*X = B,
*
*  where A is an N-by-N tridiagonal matrix, by Gaussian elimination wit
*  partial pivoting.
*
*  Note that the equation  A'*X = B  may be solved by interchanging the
*  order of the arguments DU and DL.
*
*  Arguments
*  =========
*
*  N       (input) INTEGER
*          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.
*
*  DL      (input/output) DOUBLE PRECISION array, dimension (N-1)
*          On entry, DL must contain the (n-1) subdiagonal elements of
*          A.
*          On exit, DL is overwritten by the (n-2) elements of the
*          second superdiagonal of the upper triangular matrix U from
*          the LU factorization of A, in DL(1), ..., DL(n-2).
*
*  D       (input/output) 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 U.
*
*  DU      (input/output) DOUBLE PRECISION array, dimension (N-1)
*          On entry, DU must contain the (n-1) superdiagonal elements
*          of A.
*          On exit, DU is overwritten by the (n-1) elements of the firs
*          superdiagonal of U.
*
*  B       (input/output) DOUBLE PRECISION array, dimension (LDB,NRHS)
*          On entry, the N-by-NRHS 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, and the solution
*                has not been computed.  The factorization has not been
*                completed unless i = N.
*
```

```
*  =====================================================================
*
*     .. Parameters ..
      DOUBLE PRECISION   ZERO
      PARAMETER          ( ZERO = 0.0D+0 )
*     ..
*     .. Local Scalars ..
      INTEGER            J, K
      DOUBLE PRECISION   MULT, TEMP
*     ..
*     .. Intrinsic Functions ..
      INTRINSIC          ABS, MAX
*     ..
*     .. External Subroutines ..
      EXTERNAL           XERBLA
*     ..
*     .. Executable Statements ..
*
      INFO = 0
      IF( N.LT.0 ) THEN
         INFO = -1
      ELSE IF( NRHS.LT.0 ) THEN
         INFO = -2
      ELSE IF( LDB.LT.MAX( 1, N ) ) THEN
         INFO = -7
      END IF
      IF( INFO.NE.0 ) THEN
         CALL XERBLA( 'DGTSV ', -INFO )
         RETURN
      END IF
*
      IF( N.EQ.0 )
     $   RETURN
*
```

```
      DO 30 K = 1, N - 1
         IF( DL( K ).EQ.ZERO ) THEN
*
*           Subdiagonal is zero, no elimination is required.
*
            IF( D( K ).EQ.ZERO ) THEN
*
*              Diagonal is zero: set INFO = K and return; a unique
*              solution can not be found.
*
               INFO = K
               RETURN
            END IF
         ELSE IF( ABS( D( K ) ).GE.ABS( DL( K ) ) ) THEN
*
*           No row interchange required
*
            MULT = DL( K ) / D( K )
            D( K+1 ) = D( K+1 ) - MULT*DU( K )
            DO 10 J = 1, NRHS
               B( K+1, J ) = B( K+1, J ) - MULT*B( K, J )
   10       CONTINUE
            IF( K.LT.( N-1 ) )
     $         DL( K ) = ZERO
         ELSE
*
*           Interchange rows K and K+1
*
            MULT = D( K ) / DL( K )
            D( K ) = DL( K )
            TEMP = D( K+1 )
            D( K+1 ) = DU( K ) - MULT*TEMP
            IF( K.LT.( N-1 ) ) THEN
               DL( K ) = DU( K+1 )
```

```
               DU( K+1 ) = -MULT*DL( K )
            END IF
            DU( K ) = TEMP
            DO 20 J = 1, NRHS
               TEMP = B( K, J )
               B( K, J ) = B( K+1, J )
               B( K+1, J ) = TEMP - MULT*B( K+1, J )
   20       CONTINUE
         END IF
   30 CONTINUE
      IF( D( N ).EQ.ZERO ) THEN
         INFO = N
         RETURN
      END IF
*
*     Back solve with the matrix U from the factorization.
*
      DO 50 J = 1, NRHS
         B( N, J ) = B( N, J ) / D( N )
         IF( N.GT.1 )
     $      B( N-1, J ) = ( B( N-1, J )-DU( N-1 )*B( N, J ) ) / D( N-1
         DO 40 K = N - 2, 1, -1
            B( K, J ) = ( B( K, J )-DU( K )*B( K+1, J )-DL( K )*
     $                  B( K+2, J ) ) / D( K )
   40    CONTINUE
   50 CONTINUE
*
      RETURN
*
*     End of DGTSV
*
      END
```

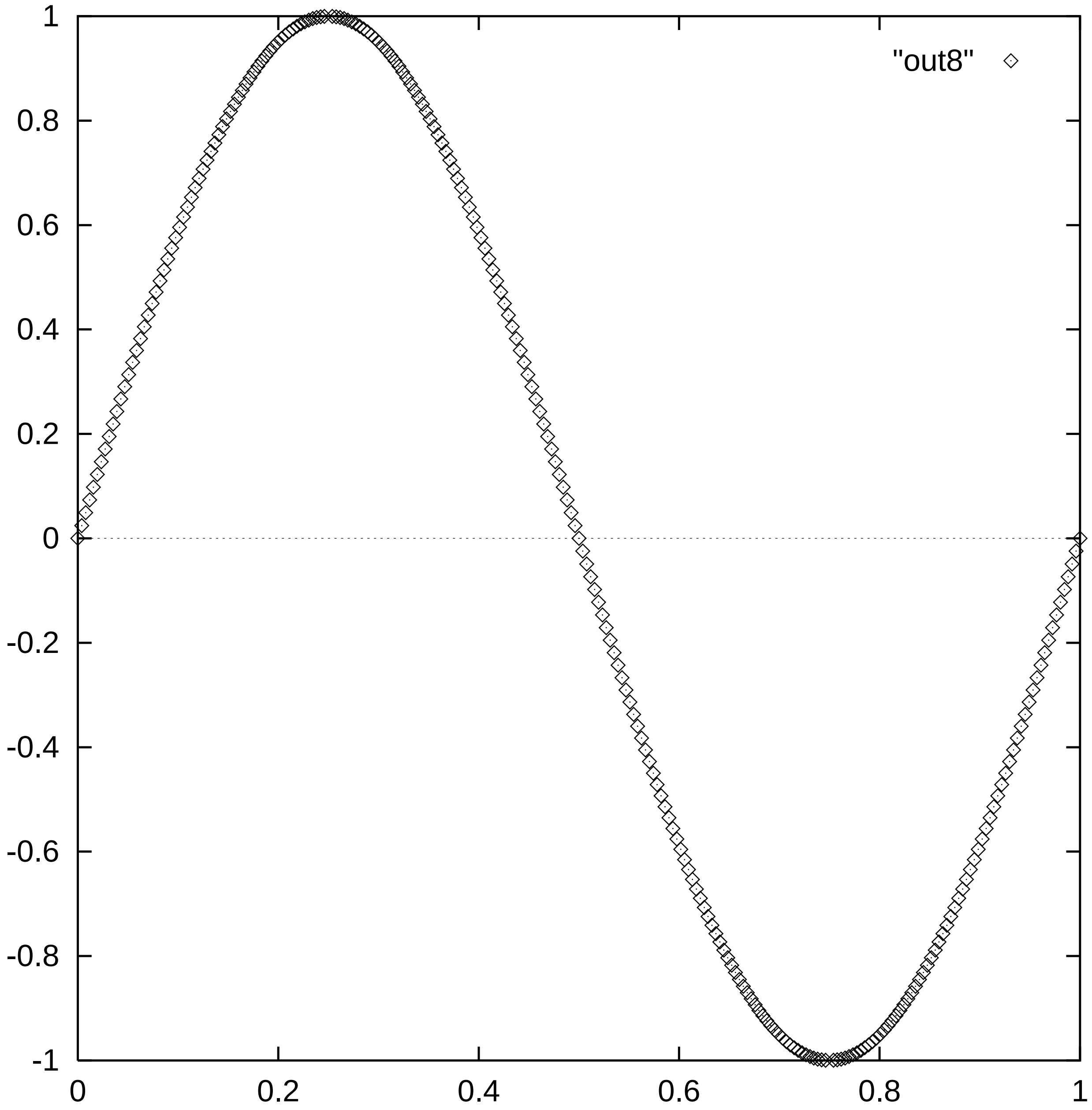
```
################################################################
# Building 'bvp1d' and sample output on sgi1.
################################################################
sgi1% pwd; ls
/usr/people/phys410/linsys/ex2
Makefile   bvp1d.f    gperr      gpsoln8

sgi1% make
f77 -g -64 -c bvp1d.f
f77 -g -64 -L/usr/local/lib bvp1d.o -lp329f -llapack -lblas -o bvp1d

sgi1% bvp1d
 usage: bvp1d <level> [<option>]

        Specify option .ne. 0 for output
        of error instead of solution

sgi1% bvp1d 4
   0.0000000000000000E+00 -5.5511151231257827E-16
   6.2500000000000000E-02  0.3876394685723090
   0.1250000000000000      0.7162643420150174
   0.1875000000000000      0.9358444623383684
   0.2500000000000000       1.012950746721879
   0.3125000000000000      0.9358444623383684
   0.3750000000000000      0.7162643420150175
   0.4375000000000000      0.3876394685723092
   0.5000000000000000     -2.2204460492503131E-16
   0.5625000000000000     -0.3876394685723097
   0.6250000000000000     -0.7162643420150181
   0.6875000000000000     -0.9358444623383690
   0.7500000000000000      -1.012950746721880
   0.8125000000000000     -0.9358444623383690
   0.8750000000000000     -0.7162643420150181
   0.9375000000000000     -0.3876394685723097
```

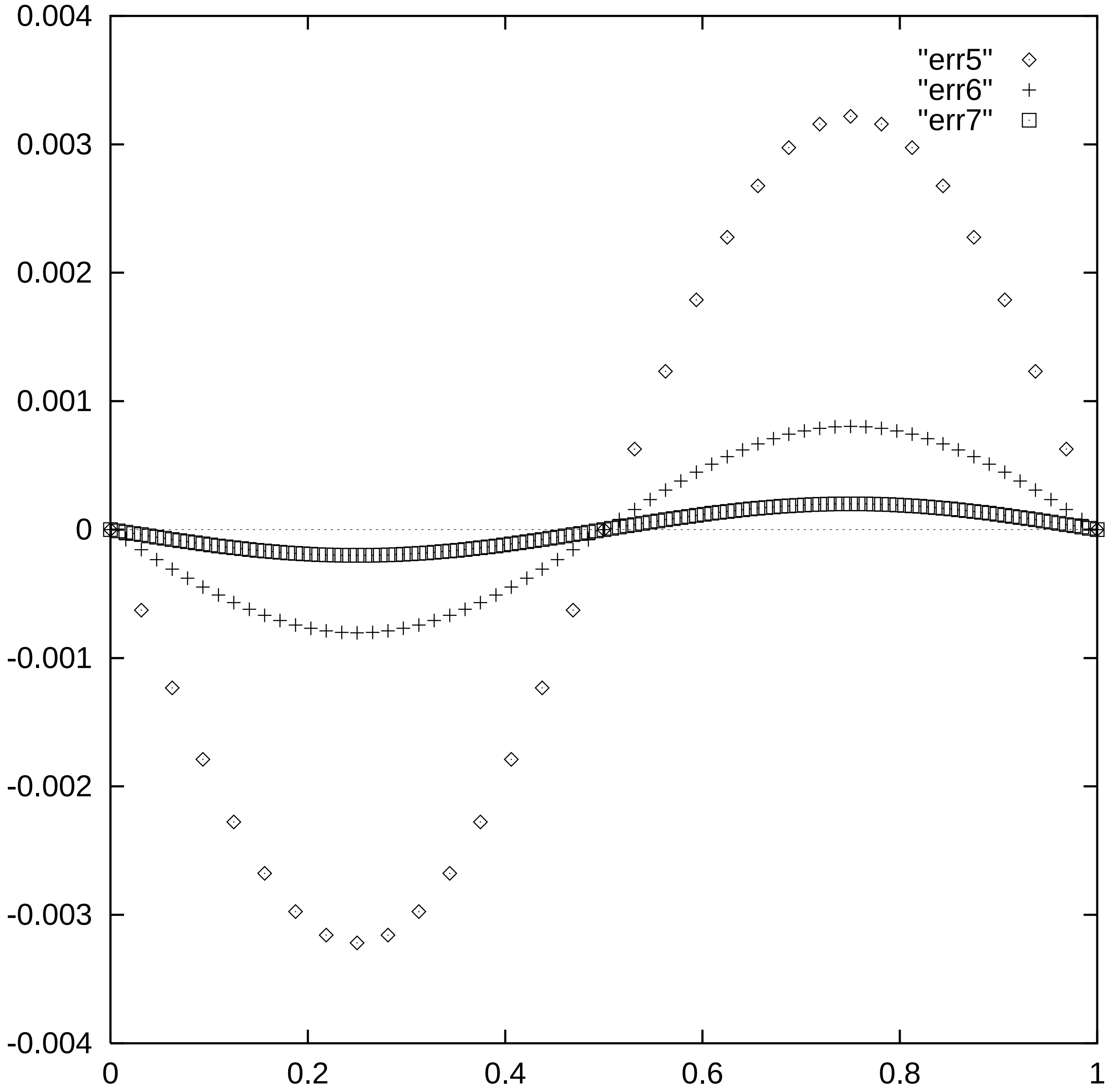
```
   1.000000000000000     -2.4492935982947064E-16
 rmserr =   8.8841389573651453E-03

#############################################################
# Simple convergence test: solve BVP on a sequence of
# levels (h, h/2, h/4, h/16, etc.), redirect stdout to
# /dev/null so that only the overall RMS error appears on
# terminal.  Note how RMS error goes down by very nearly
# a factor of 4 at each successive level, indicating
# O(h^2) convergence.
#############################################################
sgi1% foreach level (4 5 6 7 8 9 10)
foreach? bvp1d $level > /dev/null
foreach? end
 rmserr =   8.8841389573651453E-03
 rmserr =   2.2413991373367772E-03
 rmserr =   5.6382739826354859E-04
 rmserr =   1.4145099550532311E-04
 rmserr =   3.5428279660444339E-05
 rmserr =   8.8654982501522291E-06
 rmserr =   2.2174426911240527E-06

#############################################################
# Making output files for subsequent plotting via gnuplot.
# See Class Notes for postscript.
#############################################################
sgi1% bvp1d 8 > out8
 rmserr =   3.5428279660444339E-05
sgi1% bvp1d 5 1 > err5
 rmserr =   2.2413991373367772E-03
sgi1% bvp1d 6 1 > err6
 rmserr =   5.6382739826354859E-04
sgi1% bvp1d 7 1 > err7
 rmserr =   1.4145099550532311E-04
```

```
##############################################################
# Gnuplot "script" (gpsoln8) for making plot of level-8
# solution
##############################################################
sgi1% cat gpsoln8
set terminal postscript portrait
set size 0.760,1.0
set output "soln8.ps"
plot [0:1] [-1:1] "out8"
quit

##############################################################
# Make the plot
##############################################################
sgi1% gnuplot < gpsoln8

##############################################################
# Gnuplot "script" (gperr) for making plot of error from
# level 5, 6 and 7 calculations
##############################################################
sgi1% cat gperr
set terminal postscript portrait
set size 0.760,1.0
set output "err567.ps"
plot "err5", "err6", "err7"
quit

##############################################################
# Make the plot
##############################################################
sgi1% gnuplot < gperr
```

```
sgi1% ls
Makefile    bvp1d.f     err5        err6        gperr       out8
bvp1d*      bvp1d.o     err567.ps   err7        gpsoln8     soln8.ps

##############################################################
# Clean-up: Note, the Makefile used here has separate
# 'clean' and 'vclean' (very clean) targets.
##############################################################
sgi1% make clean
rm *.o
rm bvp1d

sgi1% ls
Makefile    err5        err6        gperr       out8
bvp1d.f     err567.ps   err7        gpsoln8     soln8.ps

sgi1% make vclean
rm *.o
Cannot access *.o: No such file or directory
make: [clean] Error 2 (ignored)
rm bvp1d
Cannot access bvp1d: No such file or directory
make: [clean] Error 2 (ignored)
rm err[0-9]*
rm out[0-9]*
rm *.ps

sgi1% ls
Makefile   bvp1d.f    gperr      gpsoln8
```

```
.IGNORE:

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

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

EXECUTABLES = bvp1d

all: $(EXECUTABLES)

bvp1d: bvp1d.o
   $(F77_LOAD) bvp1d.o -lp410f -llapack $(LIBBLAS) -o bvp1d

clean:
   rm *.o
   rm $(EXECUTABLES)

############################################################
# Note the 'vclean' target: 'make vclean' results in
# 'make clean' followed by removal of input and output
# data files and postscript files.
############################################################
vclean: clean
   rm err[0-9]*
   rm out[0-9]*
   rm *.ps
```


"out8"
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
0.2
0.4
0.6
0.8
1


0.004
0.003
0.002
0.001
0
-0.001
-0.002
-0.003
-0.004
0
0.2
0.4
0.6
0.8
1
"err5"
"err6"
"err7"