```
c===========================================================
c     tlsoda: Driver routine which uses ODEPACK routine
c     lsoda to solve
c
c        u''(x) = -u(x)
c===========================================================
      program         tlsoda

      implicit        none

      integer         iargc,        i4arg
      real*8          r8arg

c-----------------------------------------------------------
c     Number of ODEs (when written in canonical first order
c     form).
c-----------------------------------------------------------
      integer         neq
      parameter     ( neq = 2 )

c-----------------------------------------------------------
c     Command-line arguments (initial values of y and
c     dy/dx).
c-----------------------------------------------------------
      real*8          y0(neq)
      real*8          r8_never
      parameter     ( r8_never = -1.0d-60 )
```

```
c-----------------------------------------------------------
c     Vector for storing requested x-values for solution
c     output.
c-----------------------------------------------------------
      integer         mxnxout
      parameter     ( mxnxout = 10 000 )
      real*8          xout(mxnxout)
      integer         nxout,            ixout

c-----------------------------------------------------------
c     LSODA variables.   Note that 'fcn' and 'jac' are
c     user supplied subroutines (not functions) which
c     evaluate the RHSs of the ODEs and the Jacobian
c     of the system.  Under normal operation, the Jacobian
c     evaluator can be a dummy routine.
c-----------------------------------------------------------
      external       fcn,        jac

      real*8         y(neq)
      real*8         xbgn,       xend

      integer        itol
```

```
c---------------------------------------------------------------
c     The following comment block is extracted from the
c     LSODA documentation.
c---------------------------------------------------------------
c rtol   = relative tolerance parameter (scalar).
c atol   = absolute tolerance parameter (scalar or array).
c   the estimated local error in y(i) will be controlled so
c   as to be less than
c        ewt(i) = rtol*abs(y(i)) + atol     if itol = 1, or
c        ewt(i) = rtol*abs(y(i)) + atol(i)  if itol = 2.
c   thus the local error test passes if, in each component,
c   either the absolute error is less than atol (or atol(i)),
c   or the relative error is less than rtol.
c   use rtol = 0.0 for pure absolute error control, and
c   use atol = 0.0 (or atol(i) = 0.0) for pure relative error
c   control.  CAUTION.. actual (global) errors may exceed
c   these local tolerances, so choose them CONSERVATIVELY.
c---------------------------------------------------------------
      real*8         rtol,       atol
      integer        itask,      istate,     iopt
      integer        lrw

c---------------------------------------------------------------
c     LSODA work arrays.
c---------------------------------------------------------------
      parameter    ( lrw = 22 + neq * 16 )
      real*8         rwork(lrw)

      integer        liw
      parameter    ( liw = 20 + neq )
      integer        iwork(liw)
      integer        jt
```

```
c-----------------------------------------------------------
c     Error tolerances.
c-----------------------------------------------------------
      real*8          tol
      real*8          default_tol
      parameter     ( default_tol = 1.0d-6 )

c-----------------------------------------------------------
c     Locals.
c-----------------------------------------------------------
      integer          ieq

c-----------------------------------------------------------
c     Argument parsing.
c-----------------------------------------------------------
      if( iargc() .ne. 2 )  go to 900
      do ieq = 1 , neq
         y0(ieq) = r8arg(ieq,r8_never)
         if( y0(ieq) .eq. r8_never ) go to 900
      end do

c-----------------------------------------------------------
c     Get requested output x-values from standard input
c-----------------------------------------------------------
      call dvfrom('-',xout,nxout,mxnxout)

c-----------------------------------------------------------
c     Use 'default_tol' for all tolerances.
c-----------------------------------------------------------
      tol    = default_tol
```

```
c-----------------------------------------------------------
c     Set LSODA parameters ... see LSODA documentation
c     for fuller description.
c-----------------------------------------------------------

      itol   = 1         ! Indicates that 'atol' is scalar
      rtol   = tol       ! Relative tolerance
      atol   = tol       ! Absolute tolerance
      itask  = 1         ! Normal computation
      iopt   = 0         ! Indicates no optional inputs
      jt     = 2         ! Jacobian type

c-----------------------------------------------------------
c     Initialize the solution.
c-----------------------------------------------------------
      do ieq = 1 , neq
         y(ieq) = y0(ieq)
      end do

c-----------------------------------------------------------
c     Output initial solution.
c-----------------------------------------------------------
      write(*,*) xout(1),   y(1)
```

```
c-------------------------------------------------------------
c     Loop over requested output x-values ...
c-------------------------------------------------------------
      do ixout = 2, nxout
c-------------------------------------------------------------
c        Set interval limits of integration.
c-------------------------------------------------------------
         xbgn = xout(ixout-1)
         xend = xout(ixout)
c-------------------------------------------------------------
c        Call lsoda to integrate system on
c        x = xbgn ... xend.
c
c        'istate' should always be set to 1 prior to
c        invocation; LSODA also uses it as a return code.
c-------------------------------------------------------------
         istate = 1

         call lsoda(fcn,neq,y,xbgn,xend,
     &              itol,rtol,atol,itask,
     &              istate,iopt,rwork,lrw,iwork,liw,jac,jt)
c-------------------------------------------------------------
c        Check return code and exit with error message if
c        there was trouble.
c-------------------------------------------------------------
         if( istate .lt. 0 ) then
            write(0,1000) istate, ixout, nxout, xout(ixout-1),
     &                    xout(ixout)
1000        format(/' sode: Error return ',i2,
     &              ' from integrator LSODA.'/
     &              ' sode: At output time ',i5,' of ',i5/
     &              ' sode: Interval ',1pe11.3,' .. ',1pe11.3/)
            go to 500
         end if
```

```
c-------------------------------------------------------------
c        Output the solution.
c-------------------------------------------------------------
         write(*,*) xout(ixout),   y(1)
      end do

 500  continue

      stop

 900  continue
         write(0,*) 'usage: tlsoda <y_i> <dy/dx_i>'
      stop

      end
```

```
c===========================================================
c     Implements differential equations:
c
c     u'' = -u
c
c     y(1) := u
c     y(2) := u'
c
c     y(1)' :=  y(2)
c     y(2)' := -y(1)
c
c     Called by ODEPACK routine LSODA.
c===========================================================
      subroutine fcn(neq,x,y,yprime)
         implicit    none

         integer     neq
         real*8      x,     y(neq),    yprime(neq)

         yprime(1) =  y(2)
         yprime(2) = -y(1)

         return
      end

c===========================================================
c     Implements Jacobian (optional).  Dummy routine in
c     this case.
c===========================================================
      subroutine jac
         implicit    none

         return
      end
```

```
      subroutine lsoda (f, neq, y, t, tout, itol, rtol, atol, itask,
     1            istate, iopt, rwork, lrw, iwork, liw, jac, jt)
      external f, jac
      integer neq, itol, itask, istate, iopt, lrw, iwork, liw, jt
      double precision y, t, tout, rtol, atol, rwork
      dimension neq(1), y(1), rtol(1), atol(1), rwork(lrw), iwork(liw)
c-----------------------------------------------------------------------
c this is the march 30, 1987 version of
c lsoda.. livermore solver for ordinary differential equations, with
c         automatic method switching for stiff and nonstiff problems.
c
c this version is in double precision.
c
c lsoda solves the initial value problem for stiff or nonstiff
c systems of first order ode-s,
c     dy/dt = f(t,y) ,  or, in component form,
c     dy(i)/dt = f(i) = f(i,t,y(1),y(2),...,y(neq)) (i = 1,...,neq).
c
c this a variant version of the lsode package.
c it switches automatically between stiff and nonstiff methods.
c this means that the user does not have to determine whether the
c problem is stiff or not, and the solver will automatically choose the
c appropriate method.  it always starts with the nonstiff method.
c
c authors..
c                linda r. petzold  and  alan c. hindmarsh,
c                computing and mathematics research division, l-316
c                lawrence livermore national laboratory
c                livermore, ca 94550.
c
c references..
c 1.  alan c. hindmarsh,  odepack, a systematized collection of ode
c     solvers, in scientific computing, r. s. stepleman et al. (eds.),
c     north-holland, amsterdam, 1983, pp. 55-64.
```

```
c 2.  linda r. petzold, automatic selection of methods for solving
c     stiff and nonstiff systems of ordinary differential equations,
c     siam j. sci. stat. comput. 4 (1983), pp. 136-148.
c-----------------------------------------------------------------------
c summary of usage.
c
```

.

.

.

```
.IGNORE:

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

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

EXECUTABLES = tlsoda

all: $(EXECUTABLES)

# Note that usage of 'odepack' library also requires linking to
# 'linpack' library (LINPACK is an antecedent of LAPACK)

tlsoda: tlsoda.o dvfrom.o
   $(F77_LOAD) tlsoda.o dvfrom.o -lp410f \
               -lodepack -llinpack $(LIBBLAS) -o tlsoda

clean:
   rm *.o
   rm $(EXECUTABLES)
```

```
################################################################
# Building 'tlsoda' and sample output on sgi1.
################################################################
sgi1% pwd; ls
/usr/people/phys410/ode/ex1
Makefile   dvfrom.f   tlsoda.f

################################################################
#  NOTE: 'lsoda' uses routines from a predecessor of LAPACK
#  called LINPACK as well as BLAS routines.  All three of the
#  libraries: 'odepack', 'linpack', and 'blas' must be
#  specified at load time.
################################################################

sgi1% make
f77 -g -64 -c tlsoda.f
f77 -g -64 -c dvfrom.f
f77 -g -64 -L/usr/local/lib tlsoda.o dvfrom.o -lp410f \
               -lodepack -llinpack -lblas -o tlsoda

sgi1% tlsoda
 usage: tlsoda <y_i> <dy/dx_i>
```

```
##############################################################
# Generate a uniform-mesh of x output-values using 'dvmesh'
# and pipe it into tlsoda with initial conditions
# u(0) = 0, u'(0) = 1  --> u(x) = sin(x)
##############################################################

sgi1% dvmesh 0.0 1.0 11 | tlsoda 0.0 1.0
   0.0000000000000000E+00  0.0000000000000000E+00
   0.1000000000000000      9.9833904829367021E-02
   0.2000000000000000      0.1986703509344916
   0.3000000000000000      0.2955222777237400
   0.4000000000000000      0.3894213005717221
   0.5000000000000000      0.4794291455964967
   0.6000000000000000      0.5646464181480721
   0.7000000000000000      0.6442215920888248
   0.7999999999999999      0.7173595216370443
   0.9000000000000000      0.7833295096871006
    1.000000000000000      0.8414729721292930
```

```
#############################################################
# Use 'nf' to compute error in solution.  Note that, at least
# at the initial time, the error in the solution is roughly
# within the specified tolerance of 1.0d-6.
#############################################################

sgi1% dvmesh 0.0 1.0 11 | tlsoda 0.0 1.0 | nf _1 'sin(_1) - _2'
0.0000000000000000E+00   0
0.1000000000000000   -4.88182538865845e-07
0.2000000000000000   -1.02013943037438e-06
0.3000000000000000   -2.07106240046606e-06
0.4000000000000000   -2.95826307156188e-06
0.5000000000000000   -3.606992293681e-06
0.6000000000000000   -3.94475303677755e-06
0.7000000000000000   -3.90485113377359e-06
0.7999999999999999   -3.43073752162582e-06
0.9000000000000000   -2.60005961716381e-06
1.000000000000000   -1.98732139644608e-06
```