```fortran
c===========================================================
c     History: sode.f
c
c     Driver routine which integrates ODEs defining
c     model for deuteron.
c
c     See class notes and Arfken, Math. Methods for
c     Physicists, 2nd Edition, section 9.1.2
c     for more details.
c===========================================================
      program        deut

      implicit       none

      character*4    cdnm
      parameter    ( cdnm = 'deut' )

      integer        iargc,         indlnb,         i4arg
      real*8         r8arg

      real*8         r8_never
      parameter    ( r8_never = -1.0d-60 )

c-----------------------------------------------------------
c     Order of system.
c-----------------------------------------------------------
      integer        neq
      parameter    ( neq = 2 )

c-----------------------------------------------------------
c     Storage for solution at requested output radii.
c-----------------------------------------------------------
      integer        maxout
      parameter    ( maxout = 10 000 )

      real*8         y0(neq)
      real*8         vxout(maxout), vyout(maxout,neq),
     &               work(maxout)
      integer        nxout,         ixout,       nxout_succ
```

```
      integer        ieq

      logical        ltrace
      parameter    ( ltrace = .true. )

      integer        maxdump
      parameter    ( maxdump = 50 )

      logical        lsodatrace
      parameter    ( lsodatrace = .false. )

      logical        ivs_ok

c------------------------------------------------------------
c     LSODA Variables.
c------------------------------------------------------------
      external       fcn,        jac

      real*8         y(neq)
      real*8         tbgn,       tend
      integer        itol
      real*8         rtol,       atol
      integer        itask,      istate,     iopt
      integer        lrw

      parameter    ( lrw = 22 + neq * 16 )
      real*8         rwork(lrw)

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

      real*8         tol
      real*8         default_tol
      parameter    ( default_tol = 1.0d-8 )

c------------------------------------------------------------
c     Common communication with routine 'fcn' in 'fcn.f' ...
c------------------------------------------------------------
```

```
      include      'fcn.inc'

c-----------------------------------------------------------
c     Parse command line arguments.  Deviation from
c     'sode'.  Usage is
c
c     deut <x0> <E> [<tol>]
c-----------------------------------------------------------
      if( iargc() .lt. 3 ) go to 900

      x0      = r8arg(1,r8_never)
      E       = r8arg(2,r8_never)
      tol     = r8arg(3,default_tol)
      if( x0 .eq. r8_never   .or.  E .eq. r8_never )
     &    go to 900

c-----------------------------------------------------------
c     Both boundary conditions are fixed in this case.
c-----------------------------------------------------------
      y0(1) = 0.0d0
      y0(2) = 1.0d0

c-----------------------------------------------------------
c     Echo command line arguments if "local tracing"
c     enabled ...
c-----------------------------------------------------------
      if( ltrace ) then
         write(0,*) 'x0:  ', x0
         write(0,*) 'E:   ', E
         write(0,*) 'tol: ', tol
      end if

c-----------------------------------------------------------
c     Get output radii from standard input ...
c-----------------------------------------------------------
      call dvfrom('-',vxout,nxout,maxout)
      if( nxout .le. 0 ) then
         write(0,*)
         write(0,*) cdnm//': No output radii read from standard input'
         write(0,*) cdnm//': Use (e.g.) dvmesh or dvgmesh to '//
```

```
     &              'generate output radii.'
         write(0,*)
         write(0,*) cdnm//': Sample usages:'
         write(0,*)
         write(0,*) cdnm//': dvmesh  0.0 1.0 101'
         write(0,*) cdnm//': dvgmesh 0.0 1.0 101 10'
         write(0,*)
         stop
      end if
      if( ltrace ) then
         if( nxout .le. maxdump ) then
            call dvdump(vxout,nxout,cdnm//': output radii',0)
         else
            call dvdmp1(vxout,work,
     &                  int(1.0d0 * nxout / maxdump + 0.5d0),
     &                  nxout,cdnm//': selected output radii',0)
         end if
         write(0,*)
         write(0,*) cdnm//': Initial time: ', vxout(1)
         write(0,*)
      end if

c-------------------------------------------------------------
c     Set LSODA parameters ...
c-------------------------------------------------------------

      itol   = 1
      rtol   = tol
      atol   = tol
      itask  = 1
      iopt   = 0
      jt     = 2

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

```
c-------------------------------------------------------------
c     Do the integration ...
c-------------------------------------------------------------
      do ixout = 2 , nxout
         istate = 1
c-------------------------------------------------------------
c        Need these temporaries since lsoda overwrites
c        tend ...
c-------------------------------------------------------------
         tbgn = vxout(ixout-1)
         tend = vxout(ixout)
         call lsoda(fcn,neq,y,tbgn,tend,
     &              itol,rtol,atol,itask,
     &              istate,iopt,rwork,lrw,iwork,liw,jac,jt)
         if( lsodatrace ) then
            write(0,1000) cdnm, ixout, nxout, vxout(ixout),
     &      vxout(ixout+1)
1000        format(' ',a,': Step ',i4,'  t = ',1pe10.3,
     &      ' .. ',1pe10.3)
            write(0,*) cdnm//': lsoda reurns ', istate
         end if

         if( istate .lt. 0 ) then
            write(0,1500) cdnm, istate, ixout, nxout,
     &                    vxout(ixout-1), vxout(ixout)
1500        format(/' ',a,': Error return ',i2,
     &              ' from integrator LSODA.'/
     &              '        At output time ',i5,' of ',i5/
     &              '        Interval ',1pe11.3,' .. ',
     &             1pe11.3/)
             nxout_succ = ixout - 1
            go to 500
         end if
         do ieq = 1 , neq
            vyout(ixout,ieq) = y(ieq)
         end do
      end do
      nxout_succ = nxout
```

```
500   continue

      do ixout = 1 , nxout_succ
         write(*,2000)   vxout(ixout),
     &                 ( vyout(ixout,ieq) , ieq = 1 , neq )
c-------------------------------------------------------------
c        NOTE: This format will not dump all values on a
c        single line if neq > 10 !!
c-------------------------------------------------------------
2000     format(1P,10E25.16)
      end do

      stop

900   continue
         write(0,*) 'usage: '//cdnm//
     &              ' <x0> <E> [<tol>]'
         write(0,*) ' '
         write(0,*) '       Output radii read from standard input'
      stop

      end
```

```
c-----------------------------------------------------------
c     Driver routine which integrates ODEs defining
c     model for deuteron.
c
c     See class notes and Arfken, Math. Methods for
c     Physicists, 2nd Edition, section 9.1.2
c     for more details.
c-----------------------------------------------------------
c     neq = 2
c-----------------------------------------------------------

      subroutine fcn(neq,x,y,yprime)
         implicit    none

         include    'fcn.inc'

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

         real*8      u,     w

         u = y(1)
         w = y(2)
         yprime(1) = w

         if( x .le. x0 ) then
            yprime(2) = (-1.0d0 - E) * y(1)
         else
            yprime(2) = -E * y(1)
         end if

         return
      end
```

```
c-------------------------------------------------------------
c     Application specific common block for communication
c     with derivative evaluating routine 'fcn'.
c
c     x0:    Range of square potential well
c      E:    Energy (sought eigenvalue)
c-------------------------------------------------------------

      real*8
     &         x0,
     &         E
      common / com_fcn /
     &         x0,
     &         E
```

```
c---------------------------------------------------------------
c     Integrates static equations of motion for spherically
c     symmetric, general-relativistic boson star.
c
c     See class notes and Colpi et al, Phys Rev Lett,
c     vol 57, 2485--2488 for more details
c---------------------------------------------------------------
c     neq = 4
c---------------------------------------------------------------

      subroutine fcn(neq,x,y,yprime)
         implicit    none

         include    'fcn.inc'

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

         real*8      A,     M,     B,    sigma,    xi

         M     = y(1)
         B     = y(2)
         sigma = y(3)
         xi    = y(4)
```

```
      if( x .eq. 0.0d0 ) then
         A = 1.0d0
         yprime(1) = 0.0d0
         yprime(2) = 0.0d0
         yprime(3) = 0.0d0
         yprime(4) = ( (Omegasq / B - 1.0d0) * sigma -
  &                    Lambda * sigma**3 ) / 3.0d0
      else
         A = 1.0d0 / (1.0d0 - 2.0d0 * M / x)
         yprime(1) = x**2 * (
  &           0.5d0 * (Omegasq / B + 1.0d0) * sigma**2 +
  &           0.25d0 * Lambda * sigma**4 +
  &           0.5d0 * xi**2 / A
  &                           )
         yprime(2) = A * B * x * (
  &          (1.0d0 - 1.0d0 / A) / (x**2) +
  &          (Omegasq / B - 1.0d0) * sigma**2 -
  &          0.5d0 * Lambda * sigma**4 +
  &          xi**2 / A
  &                                )
         yprime(3) = xi
         yprime(4) = xi * (
  &          -2.0d0 / x - 0.5d0 * yprime(2) / B +
  &          A * (yprime(1) - M / x) / x
  &                        )  - A * (
  &          (Omegasq / B - 1.0d0) * sigma -
  &          Lambda * sigma**3      )
      end if

      return
   end
```

```
c-------------------------------------------------------------
c     Application specific common block for communication
c     with derivative evaluating routine 'fcn'.
c
c     Lambda:   Dimensionless phi^4 coupling constant
c     Omega:    Eigen-frequency
c     Omegasq:  Square of eigen-frequency
c-------------------------------------------------------------

      real*8
     &         Lambda,
     &         Omega,
     &         Omegasq
      common / com_fcn /
     &         Lambda,
     &         Omega,
     &         Omegasq
```