Ordinary Differential Integrators

Daniel Celis Garza

April 18, 2016

1 Directory Tree

`./applied_math/ode_int`

2 Dependencies

`nrtype.f08`

3 Subroutines

3.1 Runge-Kutta-Gill Integrator (RK45)

```
call rk4g(derivs, x, yi, yf, h)
```

3.1.1 Arguments

```
!==========================================================!
! Solve n first-order ODEs or n/2 second-order.            !
! Runge-Kutta 4 Gill's method                              !
! Daniel Celis Garza 24 Sept. 2015                         !
!----------------------------------------------------------!
! f(x,y,dydx) = ODEs to be solved                          !
! Let n := size(y) then for a k'th order system            !
! y(1+i*n/k:(i+1)*n/k) := i'th order derivative            !
!----------------------------------------------------------!
! Inputs:                                                  !
! derivs = derivatives                                     !
! h      = step size                                       !
! x      = independent variable @ step i                   !
! yi()   = array of dependent variables @ step i           !
!----------------------------------------------------------!
! Outputs:                                                 !
! yf() = array of dependent variables @ step i+1           !
!----------------------------------------------------------!
! Locals:                                                  !
! ys() = array of dependent variables @ stage s of step i !
! ki() = array of Runge-Kutta k/h                          !
! ho2  = h/2                                               !
! ci   = Butcher table parameters                          !
!==========================================================!
implicit none
```

```fortran
    real(dp), intent(in)          :: x, h, yi(:)
    real(dp), intent(out)         :: yf(:)
    real(dp), dimension(size(yi)) :: k1, k2, k3, k4, ys
    real(dp)                      :: ho2, c1, c2, c3, c4, c5, c6, c7
    parameter ( c1 = sqrt(2.0_dp), c2 = -0.5_dp * c1, c3 = 2.0_dp - c1, &
                c4 = 2.0_dp + c1, c5 = c2 - 0.5_dp, c6 = 0.5_dp * c3, &
                c7 = 0.5_dp * c4 )
    interface derivatives
      subroutine derivs(x, y, dydx)
        use nrtype
        implicit none
        real(dp), intent(in)  :: x, y(:)
        real(dp), intent(out) :: dydx(:)
      end subroutine derivs
    end interface derivatives
```

3.2 Runge-Kutta Cash-Karp Integrator (RKCK)

```fortran
call rkck(derivs, x, yi, yf, er, h)
```

3.2.1 Arguments

```fortran
    !===========================================================!
    ! Solve n first-order ODEs or n/2 second-order.             !
    ! Cash-Karp RK45                                            !
    ! Daniel Celis Garza 24 Sept. 2015                          !
    !-----------------------------------------------------------!
    ! f(x,y,dydx) = ODEs to be solved                           !
    ! Let n := size(y) then for a k'th order system             !
    ! y(1+i*n/k:(i+1)*n/k) := i'th order derivative             !
    !-----------------------------------------------------------!
    ! Inputs:                                                   !
    ! derivs = derivatives                                      !
    ! h      = step size                                        !
    ! x      = independent variable @ step i                    !
    ! yi()   = array of dependent variables @ step i            !
    !-----------------------------------------------------------!
    ! Outputs:                                                  !
    ! yf() = array of dependent variables @ step i+1            !
    ! er() = array of integration errors                        !
    !-----------------------------------------------------------!
    ! Locals:                                                   !
    ! ys() = array of dependent variables @ stage s of step i   !
    ! ki() = array of Runge-Kutta k/h                           !
    ! ci   = Butcher Table c-vector                             !
    ! aij  = Butcher Table A-matrix                             !
    ! bi   = Butcher Table b-vector                             !
    ! dbi  = b-b* vector difference for error calculation       !
    !===========================================================!
    implicit none
    real(dp), intent(in)          :: x, yi(:), h
    real(dp), intent(out)         :: yf(:), er(:)
```

```
real(dp), dimension(size(yi)) :: k1, k2, k3, k4, k5, k6, ys
real(dp)                      :: c2, c3, c4, c5, c6, a21, a31, a32, a41, &
                                 a42, a43, a51, a52, a53, a54, a61, a62, &
                                 a63, a64, a65, b1, b3, b4, b6, db1, db3, &
                                 db4, db5, db6
parameter ( c2 = .2_dp, c3 = .3_dp, c4 = .6_dp, c5 = 1._dp, c6 = .875_dp, &
            a21 = .2_dp, a31 = .075_dp, a32 = .225_dp, &
            a41 = .3_dp, a42 = -.9_dp, a43 = 1.2_dp, &
            a51 = -11._dp / 54._dp, a52 = 2.5_dp, a53 = -70._dp / 27._dp, &
            a54 = 35._dp / 27._dp, a61 = 1631._dp / 55296._dp, &
            a62 = 175._dp / 512._dp, a63 = 575._dp / 13824._dp, &
            a64 = 44275. / 110592._dp, a65 = 253. / 4096._dp, &
            b1 = 37._dp / 378._dp, b3 = 250._dp / 621._dp, &
            b4 = 125._dp / 594._dp, b6 = 512._dp / 1771._dp, &
            db1 = b1-2825._dp / 27648._dp, &
            db3 = b3 - 18575._dp / 48384._dp, &
            db4 = b4 - 13525._dp / 55296._dp, &
            db5 = -277._dp / 14336._dp, &
            db6 = b6 - 0.25_dp )
interface derivatives
  subroutine derivs(x, y, dydx)
    use nrtype
    implicit none
    real(dp), intent(in)  :: x, y(:)
    real(dp), intent(out) :: dydx(:)
  end subroutine derivs
end interface derivatives
```

3.3 Adaptive Runge-Kutta Cash-Karp Integrator (RKCKA)

```
call rkcka(derivs, x, y, der, h, hmin)
```

3.3.1 Arguments

```
!==============================================================!
! Basic Verlet Integrator (2nd order equations)                !
! Daniel Celis Garza 2nd Feb. 2016                             !
! This works best with constant acceleration.                  !
!--------------------------------------------------------------!
! f(x,y,dy) = ODEs to be solved                                !
! y         = positions                                        !
!--------------------------------------------------------------!
! Inputs:                                                      !
! derivs = derivatives                                         !
! h      = step size                                           !
! x      = independent variable                                !
!--------------------------------------------------------------!
! Inputs-Outputs:                                              !
! yi() = array of dependent variables @ step i                 !
! yf() = array of dependent variables @ step i+1               !
!--------------------------------------------------------------!
! Locals:                                                      !
```

```
  ! dxdy() = array of derivatives                              !
  ! ys()   = array of dependent variables for storage purposes !
  !============================================================!
  implicit none
  real(dp), intent(in)          :: x, h
  real(dp), intent(inout)       :: yi(:), yf(:)
  real(dp), dimension(size(yi)) :: dydx, ys
  real(dp)                      :: h2
  interface derivatives
    subroutine derivs(x, y, dydx)
      use nrtype
      implicit none
      real(dp), intent(in)  :: x, y(:)
      real(dp), intent(out) :: dydx(:)
    end subroutine derivs
  end interface derivatives
```

3.4 Basic Verlet

```
call basic_verlet(derivs, x, yi, yf, h)
```

3.4.1 Arguments

```
  !===================================================!
  ! Velocity Verlet Integrator (2nd order equations) !
  ! Daniel Celis Garza 2nd Feb. 2016                  !
  !---------------------------------------------------!
  ! f(x,y,dy) = ODEs to be solved                     !
  ! Let n := size(y), then y(1:n/2) := velocities     !
  ! and y(n/2+1:n) := accelerations                   !
  !---------------------------------------------------!
  ! Inputs:                                           !
  ! derivs = derivatives                              !
  ! h      = step size                                !
  ! x      = independent variable                     !
  ! yi() = array of dependent variables @ step i      !
  !---------------------------------------------------!
  ! Outputs:                                          !
  ! yf() = array of dependent variables @ step i+1    !
  !---------------------------------------------------!
  ! Locals:                                           !
  ! dxdy() = array of derivatives                     !
  !===================================================!
  implicit none
  real(dp), intent(in)            :: x, h, yi(:)
  real(dp), intent(out)           :: yf(:)
  integer                         :: n_coords, n_derivs
  real(dp), dimension(size(yi)/2) :: dydx, dydxs
  real(dp)                        :: h2
  interface derivatives
    subroutine derivs(x,y,dydx)
      use nrtype
      implicit none
      real(dp), intent(in)  :: x, y(:)
      real(dp), intent(out) :: dydx(:)
    end subroutine derivs
  end interface derivatives
```