HW4

Path to source code: `/home/d/dx/dxj4360/HW4`

P1: Integrate DE by rk2

The ODE

$\frac{dv}{dt} = \frac{P}{mv}$, integrate from $[0, 200]$ with $v_0 = 4.0 m/s$, $P = 400 Watt$, $m = 70 kg$

The exact solution is: $v = \sqrt{v_0^2 + 2Pt/m}$

We have:

$\frac{dx}{dt} = v$, where $v = y(2)$

$\frac{dv}{dt} = \omega/v$, where $\omega = \frac{P}{m} = 5.7143$

Modification of the DE function

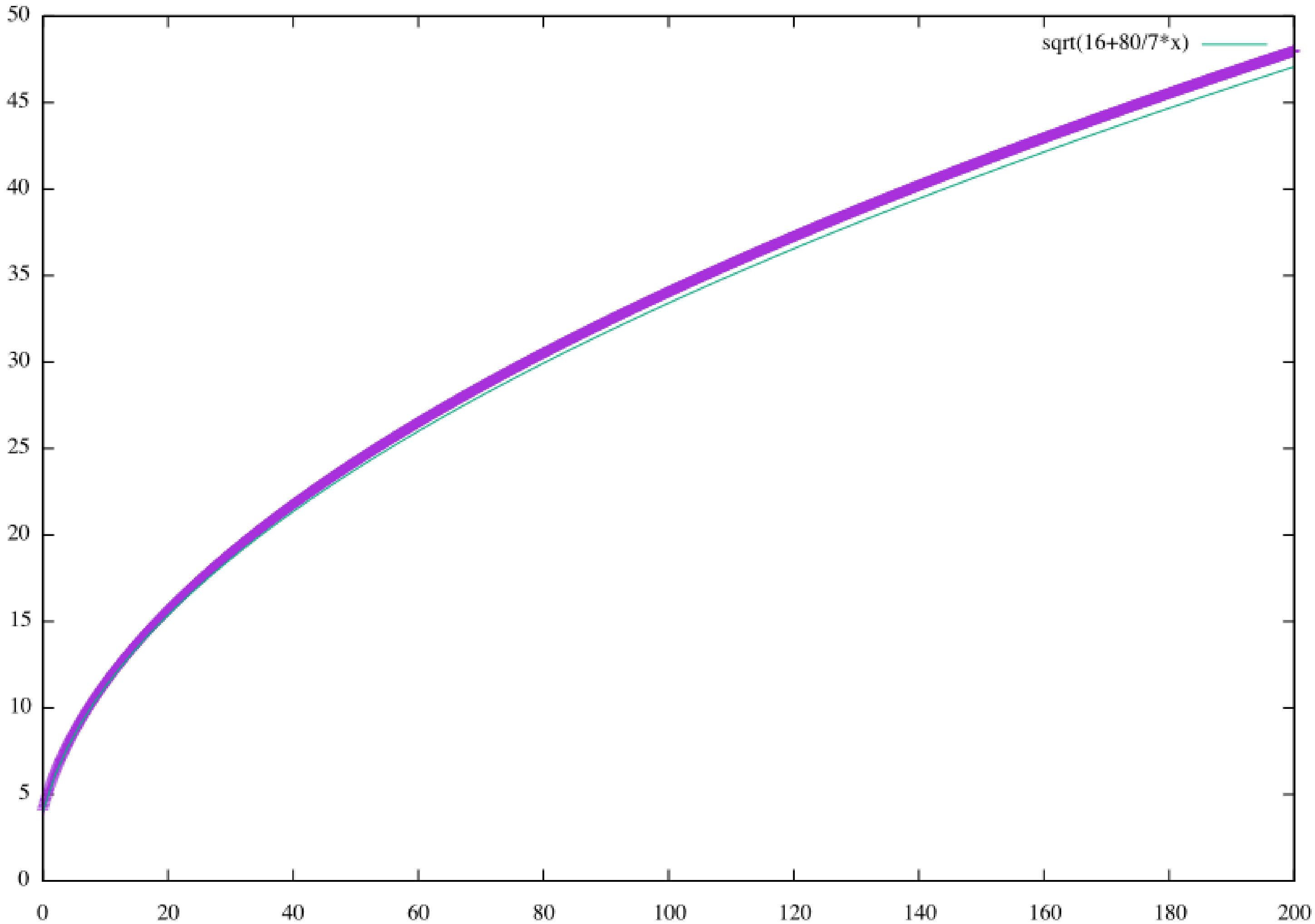
```
real*8 function deriv(temp, i)
    Implicit none
    Real*8 :: omega
    real*8, dimension(2), intent(inout):: temp
    Integer :: i
    data omega /5.7143d0/

    If (i .EQ. 1) deriv=temp(2)
    If (i .EQ. 2) deriv=omega / temp(2)
    Return
End function deriv
```

Result of the Integration

The final result is

```
 0.0000000000000000        4.1403512219144201
0.10000000000000001        4.2761034937351496
0.20000000000000001        4.4076808606527766
0.30000000000000004        4.5354460088889015
0.40000000000000002        4.6597120097890157
0.50000000000000000        4.7807513250099660
0.59999999999999998        4.8988028147021838
0.69999999999999996        5.0140772643314993
0.79999999999999993        5.1267617955466305
0.89999999999999991        5.2370234245831986
...
 199.09999999999297        47.880809961041173
 199.19999999999297        47.892742900230651
 199.29999999999296        47.904672866953852
 199.39999999999296        47.916599863430982
 199.49999999999295        47.928523891879465
 199.59999999999295        47.940444954513978
 199.69999999999294        47.952363053546449
 199.79999999999293        47.964278191186054
 199.89999999999293        47.976190369639220
 199.99999999999292        47.988099591109645
```



The green line is the exact solution. My result using rk2 is acceptable.

P2: with additional air-grad term

The ODE

$\frac{dv}{dt} = \frac{P}{mv} - Av^2 = \omega/v - \alpha v^2$, where $\omega = \frac{P}{m} = 5.7143$ and $\alpha = -0.15$

Again:

$\frac{dx}{dt} = v$

$\frac{dv}{dt} = \omega/v - \alpha v^2$

Modification of the DE function

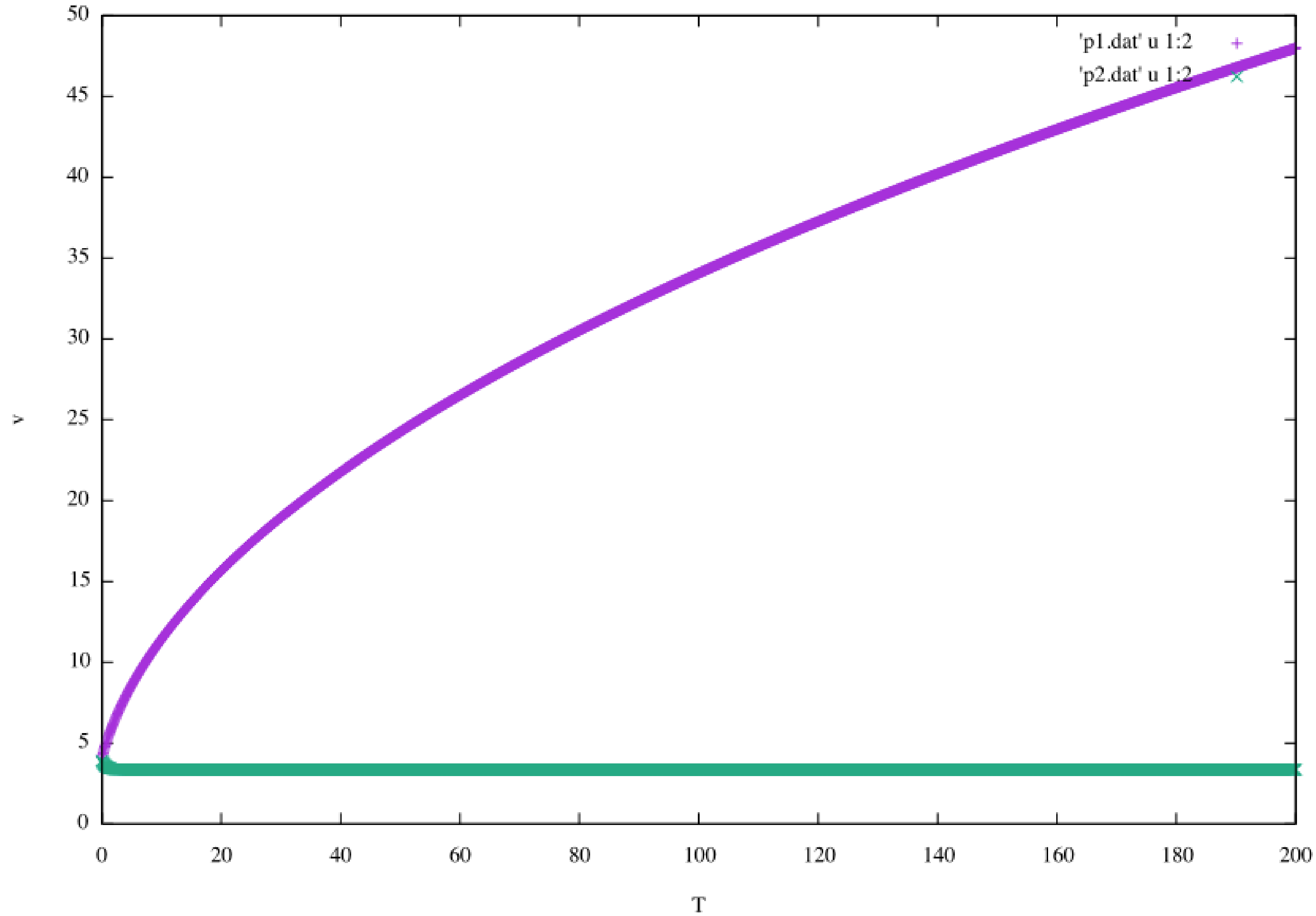
```
real*8 function deriv(temp, i)
    Implicit none
    Real*8 :: omega, alpha
    real*8, dimension(2), intent(inout):: temp
    Integer :: i
    data omega /5.7143d0/
    data alpha /0.15d0/

    If (i .EQ. 1) deriv=temp(2)
    If (i .EQ. 2) deriv=omega / temp(2) - alpha* temp(2)**2
    Return
End function deriv
```

Result of the Integration, compared to P1

The final result is

```
  0.0000000000000000        3.9104066773002248
 0.10000000000000001        3.8335965544074022
 0.20000000000000001        3.7676949971424962
 0.30000000000000004        3.7111194936880341
 0.40000000000000002        3.6625285040563833
 0.50000000000000000        3.6207809805232194
 0.59999999999999998        3.5849038738538459
 0.69999999999999996        3.5540657176454782
 0.79999999999999993        3.5275549167470794
 0.89999999999999991        3.5047617352693576
 ...
  199.09999999999297        3.3647845354623893
  199.19999999999297        3.3647845354623893
  199.29999999999296        3.3647845354623893
  199.39999999999296        3.3647845354623893
  199.49999999999295        3.3647845354623893
  199.59999999999295        3.3647845354623893
  199.69999999999294        3.3647845354623893
  199.79999999999293        3.3647845354623893
  199.89999999999293        3.3647845354623893
  199.99999999999292        3.3647845354623893
```



Dramatic difference because of the air-grad term.

P3: 4th Runge-Kutta for damped oscillation problem

The ODE

$\frac{d^2x}{dt^2} = -\omega^2 x - \alpha \frac{dx}{dt} = -\omega^2 x - \alpha v$, where $\omega = 3.0$ and $\alpha = 0.5$

Here:

$\frac{dx}{dt} = v$

$\frac{d^2x}{dt^2} = -\omega^2 x - \alpha v$, where $x = y(1)$

Modification of the DE function

```
Function deriv(x, temp, i)
    Implicit none
    Real*8 deriv, x, temp(2), omega, alpha
    Integer i
    data omega /3.0d0/
    data alpha /0.5d0/
    If (i .EQ. 1) deriv=temp(2)
    If (i .EQ. 2) deriv= - omega**2 * temp(1) - alpha * temp(2)
Return
```

rk4 iteration

$k_0 = hf(x_n, y_n)$

$k_1 = hf(x_n + 1/2h, y_n + 1/2k_0)$

$k_2 = hf(x_n + 1/2h, y_n + 1/2k_1)$

$k_3 = hf(x_n + h, y_n + k_2)$

$y_{(n+1)} = y_n + 1/6(k_0 + 2k_1 + 2k_2 + k_3) + O(h^5)$

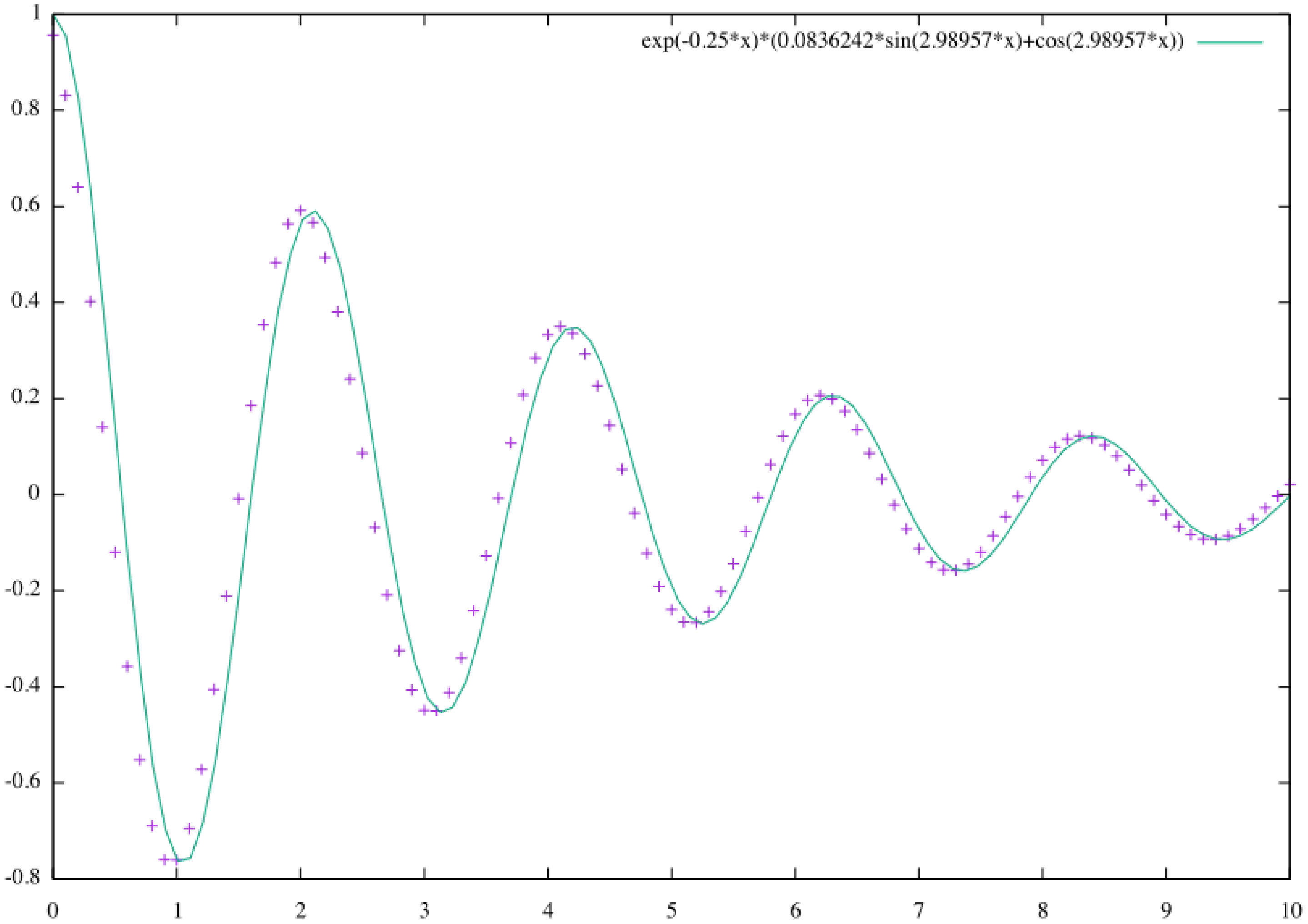
For simplicity, I seperate the `deriv` function into two functions: $f = \frac{dx}{dt} = v$ and $g = \frac{d^2x}{dt^2} = a$. Thus, I introduce parameter l together with k to calculate the two parts.

```fortran
Subroutine rk4(t, dt, y, n)
    Implicit none
    real*8, external :: df, dg
    real*8 :: h, k0, k1, k2, k3, l0, l1, l2, l3
    real*8, intent(in) :: t, dt
    real*8, intent(inout), dimension(2) :: y
    integer, intent(in) :: n

    h=dt/2.0D0
    k0 = dt * df(y(1),y(n))
    l0 = dt * dg(y(1),y(n))
    k1 = dt * df(y(1)+h, y(n)+0.5d0*l0)
    l1 = dt * dg(y(1)+0.5d0*k0,y(n)+0.5d0*l0)
    k2 = dt * df(y(1)+0.5d0*k1,y(n)+0.5d0*l1)
    l2 = dt * dg(y(1)+0.5d0*k1,y(n)+0.5d0*l1)
    k3 = dt * df(y(1)+k2,y(n)+l2)
    l3 = dt * dg(y(1)+k2,y(n)+l2)
    y(1) = y(1) + (k0+2*k1+2*k2+k3)/6.0d0
    y(n) = y(n) + (l0+2*l1+2*l2+l3)/6.0d0
    Return
End subroutine rk4
```

Result of x vs t

The approximated solution: $x(t) = e^{-0.25t}(0.0836242 * \sin(2.98957t) + \cos(2.98957t))$



The points are results from rk4 and the green curve is the approximated solution from [Wolfram Alpha](#).

Appendix for code

P1

```
Program P1
    Implicit none
    real*8 :: dt, min, max, t
    real*8, dimension(2) :: y
    integer :: n
    parameter (n=2) ! y 1: pos; 2: vel;

    ! inteval
    min=0.0D0
    max=200.0D0
    ! time step
```

```fortran
    dt=0.1D0
    ! initial position
    y(1)=0.0D0
    ! initial velocity
    y(2)=4.0D0

    ! print the exact result
    print *, sqrt(4.0D0**2.D0+2.d0*400*max/70)
    Open(6, File='P1.dat')

    ! Runga-Kutta iteration
    Do t=min, max, dt
        Call rk2(t, dt, y, n)
        Write (6,*) t, y(2) ! y(2): vel
    enddo

    Close(6)
End program P1

! second-order Runge-Kutta subroutine
! will update y(1) & y(2) based on n
subroutine rk2(t, dt, y, n)
    Implicit none
    real*8, external :: deriv
    real*8, intent(in) :: t, dt
    real*8, intent(inout), dimension(2) :: y
    real*8 :: h
    real*8, dimension(2) :: k1, k2, t1
    integer :: i,n

    h=dt/2.0D0
    Do i = 1,n
       k1(i) = dt * deriv(y, i)
       t1(i) = y(i) + 0.5D0*k1(i)
    enddo
    Do i = 1,n
       k2(i) = dt * deriv(t1, i)
       y(i) = y(i) + k2(i)
    enddo
    Return
End subroutine rk2

! function which returns the derivatives (RHS)
real*8 function deriv(temp, i)
    Implicit none
```

```
! declarations
    Real*8 :: omega
    real*8, dimension(2), intent(inout):: temp
    Integer :: i
    data omega /5.7143d0/
    !   dx/dt=v
    !   dv/dt=P/(mv)= 400/70/v=5.7143/v
    If (i .EQ. 1) deriv=temp(2)
    If (i .EQ. 2) deriv=omega / temp(2)
    Return
End function deriv
```

P2

```
Program P2
    Implicit none
    real*8 :: dt, min, max, t
    real*8, dimension(2) :: y
    integer :: n
    parameter (n=2) ! y 1: pos; 2: vel;

    ! inteval
    min=0.0D0
    max=200.0D0
    ! time step
    dt=0.1D0
    ! initial position
    y(1)=0.0D0
    ! initial velocity
    y(2)=4.0D0

    Open(6, File='P2.dat')
    ! Runga-Kutta iteration
    Do t=min, max, dt
        Call rk2(t, dt, y, n)
        Write (6,*) t, y(2) ! y(2): vel
    enddo

    Close(6)
End program P2

! second-order Runge-Kutta subroutine
! will update y(1) & y(2) based on n
subroutine rk2(t, dt, y, n)
```

```
    Implicit none
    real*8, external :: deriv
    real*8, intent(in) :: t, dt
    real*8, intent(inout), dimension(2) :: y
    real*8 :: h
    real*8, dimension(2) :: k1, k2, t1
    integer :: i,n

    h=dt/2.0D0
    Do i = 1,n
       k1(i) = dt * deriv(y, i)
       t1(i) = y(i) + 0.5D0*k1(i)
    enddo
    Do i = 1,n
       k2(i) = dt * deriv(t1, i)
       y(i) = y(i) + k2(i)
    enddo
    Return
End subroutine rk2

! function which returns the derivatives (RHS)
real*8 function deriv(temp, i)
    Implicit none
! declarations
    Real*8 :: omega, alpha
    real*8, dimension(2), intent(inout):: temp
    Integer :: i
    data omega /5.7143d0/
    data alpha /0.15d0/

    If (i .EQ. 1) deriv=temp(2)
    If (i .EQ. 2) deriv=omega / temp(2) - alpha* temp(2)**2
    Return
End function deriv
```

P3

```
Program P3
    Implicit none
    real*8 :: dt, min, max, t
    real*8, dimension(2) :: y

    ! inteval
    min=0.0D0
```

```fortran
    max=10.0D0

    !time step
    dt=0.1D0
    ! initial position
    y(1)=1.0D0
    ! initial velocity
    y(2)=0.0D0

    Open(6, File='p3.dat')
    ! Runga-Kutta iteration
    Do t=min, max, dt
        Call rk4(t, dt, y, 2)
        Write (6,*) t, y(1) ! y(1): x, pos
    enddo

    Close(6)
End program P3

! 4th-order Runge-Kutta subroutine
Subroutine rk4(t, dt, y, n)
    Implicit none
    real*8, external :: df, dg
    real*8 :: h, k0, k1, k2, k3, l0, l1, l2, l3
    real*8, intent(in) :: t, dt
    real*8, intent(inout), dimension(2) :: y
    integer, intent(in) :: n

    h=dt/2.0D0
    k0 = dt * df(y(1),y(n))
    l0 = dt * dg(y(1),y(n))
    k1 = dt * df(y(1)+h, y(n)+0.5d0*l0)
    l1 = dt * dg(y(1)+0.5d0*k0,y(n)+0.5d0*l0)
    k2 = dt * df(y(1)+0.5d0*k1,y(n)+0.5d0*l1)
    l2 = dt * dg(y(1)+0.5d0*k1,y(n)+0.5d0*l1)
    k3 = dt * df(y(1)+k2,y(n)+l2)
    l3 = dt * dg(y(1)+k2,y(n)+l2)
    y(1) = y(1) + (k0+2*k1+2*k2+k3)/6.0d0
    y(n) = y(n) + (l0+2*l1+2*l2+l3)/6.0d0
    Return
End subroutine rk4

! function which returns the derivatives (RHS)
real*8 function df(y,z)
! dx/dt = v(t)
```

```
    Implicit none
    real*8 ,intent(in) :: y, z
    df=z
    Return
End function df

real*8 Function dg(y,z)
! dv/dt = -w**2*x(t) -a*v
! the second term is damping term
    implicit none
    real*8, intent(in) :: y, z
    real*8 :: omega, alpha
    data omega /3.0d0/
    data alpha /0.5d0/

    dg= - omega**2.0d0 * y - alpha * z
    Return
End function dg
```