

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 $v_0 = 4.0m/s$, $P = 400Watt$, $m = 70kg$

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 $\omega = \frac{P}{m} = 5.7143$

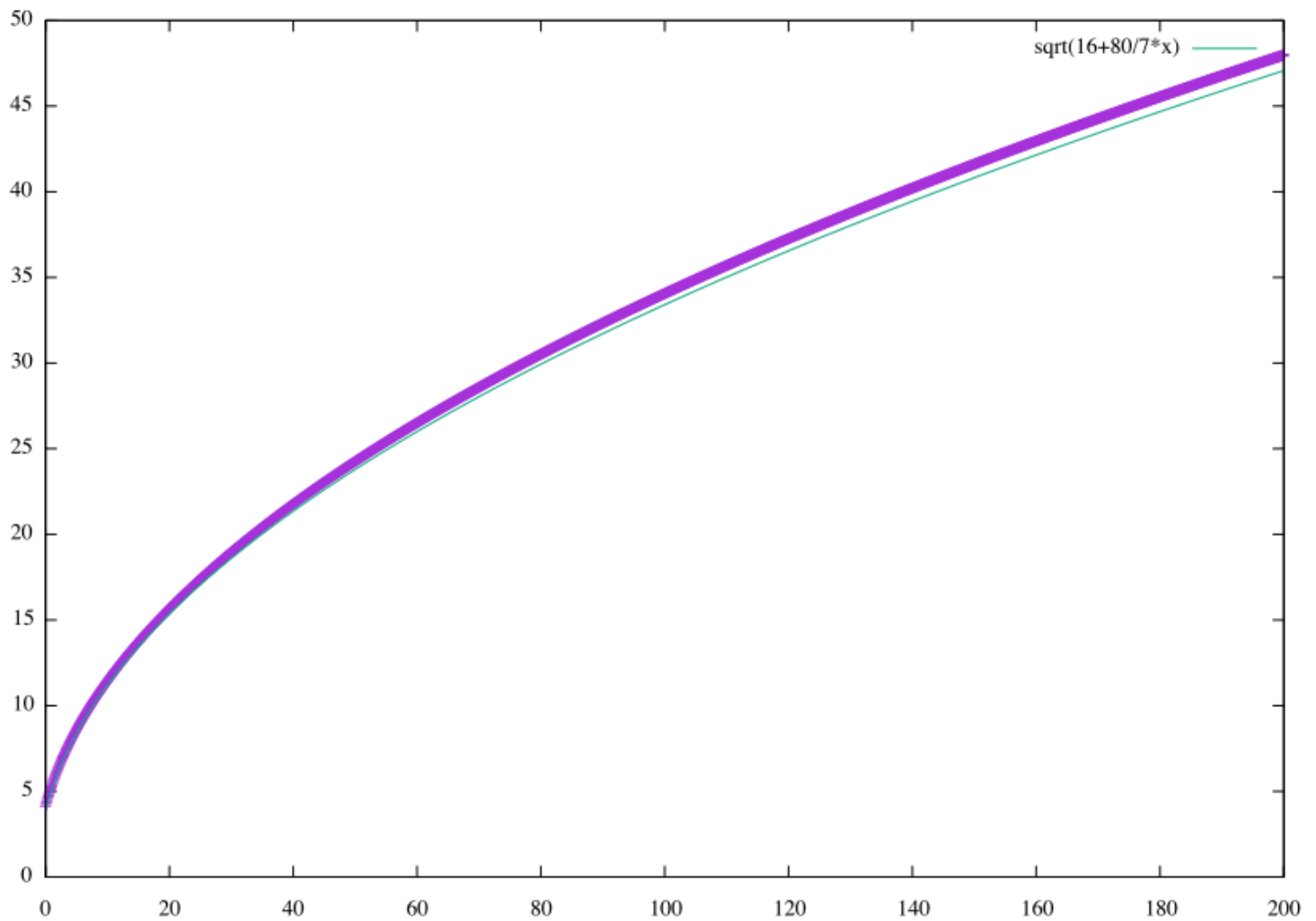
Modification of the DE function

```
1  real*8 function deriv(temp, i)
2      Implicit none
3      Real*8 :: omega
4      real*8, dimension(2), intent(inout):: temp
5      Integer :: i
6      data omega /5.7143d0/
7
8      If (i .EQ. 1) deriv=temp(2)
9      If (i .EQ. 2) deriv=omega / temp(2)
10     Return
11 End function deriv
```

Result of the Integration

The final result is

1	0.0000000000000000	4.1403512219144201
2	0.10000000000000001	4.2761034937351496
3	0.20000000000000001	4.4076808606527766
4	0.30000000000000004	4.5354460088889015
5	0.40000000000000002	4.6597120097890157
6	0.5000000000000000	4.7807513250099660
7	0.5999999999999998	4.8988028147021838
8	0.6999999999999996	5.0140772643314993
9	0.7999999999999993	5.1267617955466305
10	0.8999999999999991	5.2370234245831986
11	...	
12	199.0999999999297	47.880809961041173
13	199.1999999999297	47.892742900230651
14	199.2999999999296	47.904672866953852
15	199.3999999999296	47.916599863430982
16	199.4999999999295	47.928523891879465
17	199.5999999999295	47.940444954513978
18	199.6999999999294	47.952363053546449
19	199.7999999999293	47.964278191186054
20	199.8999999999293	47.976190369639220
21	199.9999999999292	47.988099591109645



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

P2: with additional air-grad term

The ODE

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

Again:

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

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

Modification of the DE function

```

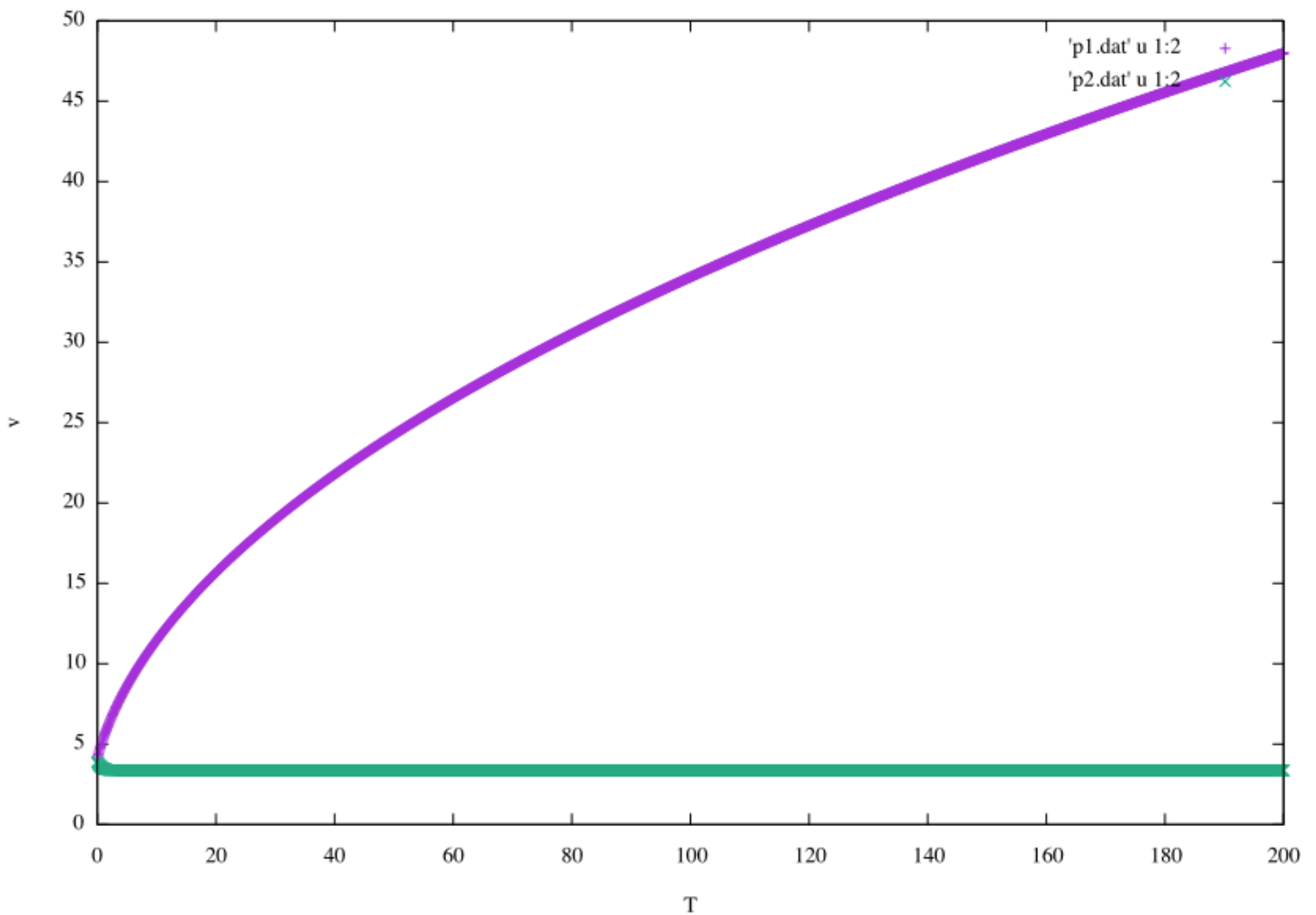
1  real*8 function deriv(temp, i)
2      Implicit none
3      Real*8 :: omega, alpha
4      real*8, dimension(2), intent(inout):: temp
5      Integer :: i
6      data omega /5.7143d0/
7      data alpha /0.15d0/
8
9      If (i .EQ. 1) deriv=temp(2)
10     If (i .EQ. 2) deriv=omega / temp(2) - alpha* temp(2)**2
11     Return
12 End function deriv

```

Result of the Integration, compared to P1

The final result is

1	0.0000000000000000	3.9104066773002248
2	0.10000000000000001	3.8335965544074022
3	0.20000000000000001	3.7676949971424962
4	0.30000000000000004	3.7111194936880341
5	0.40000000000000002	3.6625285040563833
6	0.50000000000000000	3.6207809805232194
7	0.5999999999999998	3.5849038738538459
8	0.6999999999999996	3.5540657176454782
9	0.7999999999999993	3.5275549167470794
10	0.8999999999999991	3.5047617352693576
11	...	
12	199.09999999999297	3.3647845354623893
13	199.19999999999297	3.3647845354623893
14	199.29999999999296	3.3647845354623893
15	199.39999999999296	3.3647845354623893
16	199.49999999999295	3.3647845354623893
17	199.59999999999295	3.3647845354623893
18	199.69999999999294	3.3647845354623893
19	199.79999999999293	3.3647845354623893
20	199.89999999999293	3.3647845354623893
21	199.99999999999292	3.3647845354623893



Dramatic difference because of the air-grad term.

P3: 4th Runge-Kutta for damped oscillation problem

The ODE

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

Here:

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

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

Modification of the DE function

```

1 Function deriv(x, temp, i)
2   Implicit none
3   Real*8 deriv, x, temp(2), omega, alpha
4   Integer i
5   data omega /3.0d0/
6   data alpha /0.5d0/
7   If (i .EQ. 1) deriv=temp(2)
8   If (i .EQ. 2) deriv= - omega**2 * temp(1) - alpha * temp(2)
9   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 separate 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.

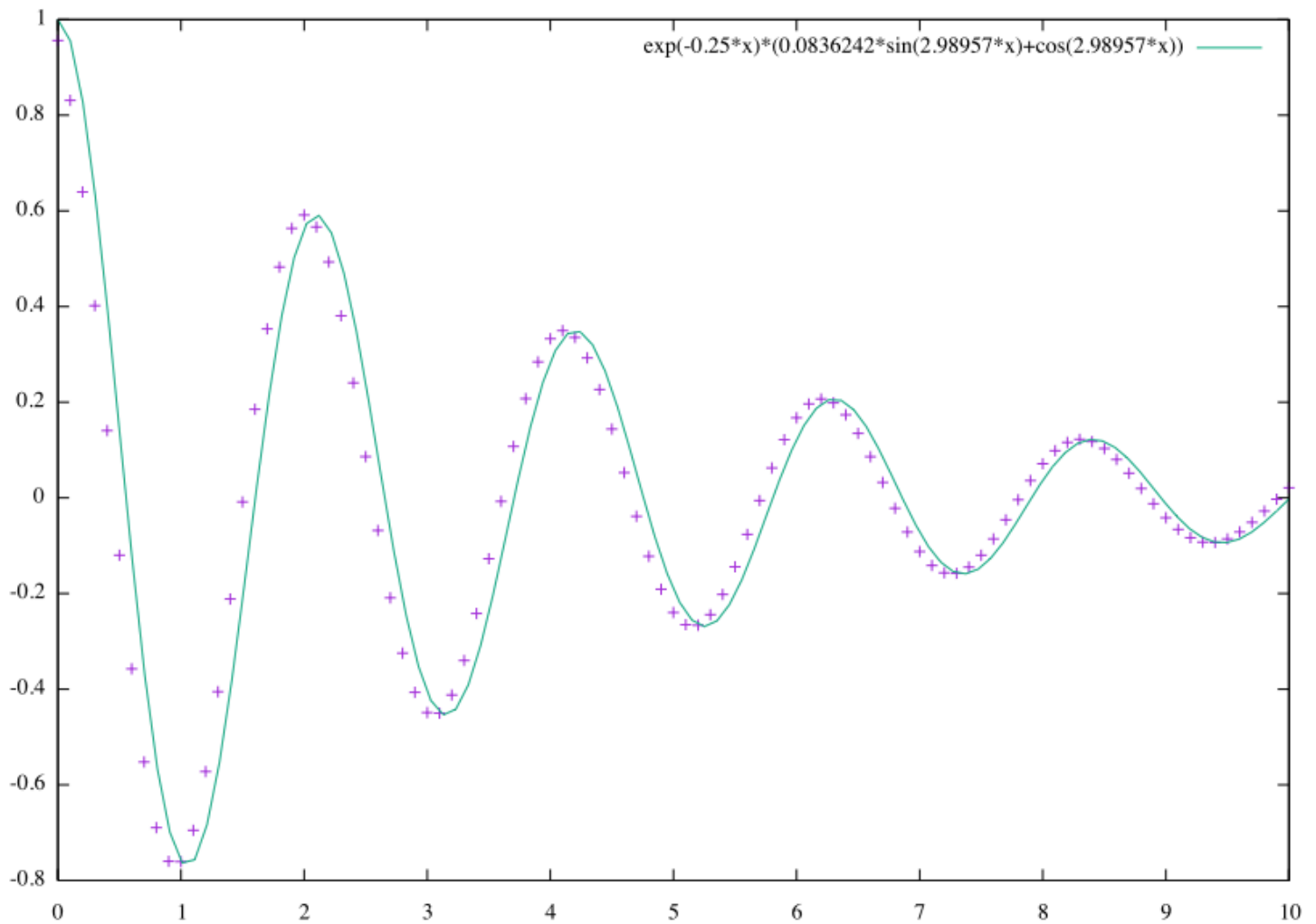
```

1 Subroutine rk4(t, dt, y, n)
2   Implicit none
3   real*8, external :: df, dg
4   real*8 :: h, k0, k1, k2, k3, l0, l1, l2, l3
5   real*8, intent(in) :: t, dt
6   real*8, intent(inout), dimension(2) :: y
7   integer, intent(in) :: n
8
9   h=dt/2.0d0
10  k0 = dt * df(y(1),y(n))
11  l0 = dt * dg(y(1),y(n))
12  k1 = dt * df(y(1)+h, y(n)+0.5d0*l0)
13  l1 = dt * dg(y(1)+0.5d0*k0,y(n)+0.5d0*l0)
14  k2 = dt * df(y(1)+0.5d0*k1,y(n)+0.5d0*l1)
15  l2 = dt * dg(y(1)+0.5d0*k1,y(n)+0.5d0*l1)
16  k3 = dt * df(y(1)+k2,y(n)+l2)
17  l3 = dt * dg(y(1)+k2,y(n)+l2)
18  y(1) = y(1) + (k0+2*k1+2*k2+k3)/6.0d0
19  y(n) = y(n) + (l0+2*l1+2*l2+l3)/6.0d0
20  Return
21 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](https://www.wolframalpha.com/).

Appendix for code

P1

```

1 Program P1
2   Implicit none
3   real*8 :: dt, min, max, t
4   real*8, dimension(2) :: y
5   integer :: n
6   parameter (n=2) ! y 1: pos; 2: vel;
7
8   ! interval
9   min=0.0D0
10  max=200.0D0
11  ! time step

```



```

12  dt=0.1D0
13  ! initial position
14  y(1)=0.0D0
15  ! initial velocity
16  y(2)=4.0D0
17
18  ! print the exact result
19  print *, sqrt(4.0D0**2.D0+2.d0*400*max/70)
20  Open(6, File='P1.dat')
21
22  ! Runga-Kutta iteration
23  Do t=min, max, dt
24      Call rk2(t, dt, y, n)
25      Write (6,*) t, y(2) ! y(2): vel
26  enddo
27
28  Close(6)
29 End program P1
30
31 ! second-order Runge-Kutta subroutine
32 ! will update y(1) & y(2) based on n
33 subroutine rk2(t, dt, y, n)
34     Implicit none
35     real*8, external :: deriv
36     real*8, intent(in) :: t, dt
37     real*8, intent(inout), dimension(2) :: y
38     real*8 :: h
39     real*8, dimension(2) :: k1, k2, t1
40     integer :: i,n
41
42     h=dt/2.0D0
43     Do i = 1,n
44         k1(i) = dt * deriv(y, i)
45         t1(i) = y(i) + 0.5D0*k1(i)
46     enddo
47     Do i = 1,n
48         k2(i) = dt * deriv(t1, i)
49         y(i) = y(i) + k2(i)
50     enddo
51     Return
52 End subroutine rk2
53
54 ! function which returns the derivatives (RHS)
55 real*8 function deriv(temp, i)
56     Implicit none

```

```

57 ! declarations
58   Real*8 :: omega
59   real*8, dimension(2), intent(inout):: temp
60   Integer :: i
61   data omega /5.7143d0/
62   !   dx/dt=v
63   !   dv/dt=P/(mv)= 400/70/v=5.7143/v
64   If (i .EQ. 1) deriv=temp(2)
65   If (i .EQ. 2) deriv=omega / temp(2)
66   Return
67 End function deriv

```

P2

```

1  Program P2
2    Implicit none
3    real*8 :: dt, min, max, t
4    real*8, dimension(2) :: y
5    integer :: n
6    parameter (n=2) ! y 1: pos; 2: vel;
7
8    ! interval
9    min=0.0D0
10   max=200.0D0
11   ! time step
12   dt=0.1D0
13   ! initial position
14   y(1)=0.0D0
15   ! initial velocity
16   y(2)=4.0D0
17
18   Open(6, File='P2.dat')
19   ! Runga-Kutta iteration
20   Do t=min, max, dt
21     Call rk2(t, dt, y, n)
22     Write (6,*) t, y(2) ! y(2): vel
23   enddo
24
25   Close(6)
26 End program P2
27
28 ! second-order Runge-Kutta subroutine
29 ! will update y(1) & y(2) based on n
30 subroutine rk2(t, dt, y, n)

```

```

31  Implicit none
32  real*8, external :: deriv
33  real*8, intent(in) :: t, dt
34  real*8, intent(inout), dimension(2) :: y
35  real*8 :: h
36  real*8, dimension(2) :: k1, k2, t1
37  integer :: i,n
38
39  h=dt/2.0D0
40  Do i = 1,n
41      k1(i) = dt * deriv(y, i)
42      t1(i) = y(i) + 0.5D0*k1(i)
43  enddo
44  Do i = 1,n
45      k2(i) = dt * deriv(t1, i)
46      y(i) = y(i) + k2(i)
47  enddo
48  Return
49  End subroutine rk2
50
51  ! function which returns the derivatives (RHS)
52  real*8 function deriv(temp, i)
53      Implicit none
54      ! declarations
55      Real*8 :: omega, alpha
56      real*8, dimension(2), intent(inout):: temp
57      Integer :: i
58      data omega /5.7143d0/
59      data alpha /0.15d0/
60
61      If (i .EQ. 1) deriv=temp(2)
62      If (i .EQ. 2) deriv=omega / temp(2) - alpha* temp(2)**2
63      Return
64  End function deriv

```

P3

```

1  Program P3
2      Implicit none
3      real*8 :: dt, min, max, t
4      real*8, dimension(2) :: y
5
6      ! interval
7      min=0.0D0

```

```

8      max=10.0D0
9
10     !time step
11     dt=0.1D0
12     ! initial position
13     y(1)=1.0D0
14     ! initial velocity
15     y(2)=0.0D0
16
17     Open(6, File='p3.dat')
18     ! Runga-Kutta iteration
19     Do t=min, max, dt
20         Call rk4(t, dt, y, 2)
21         Write (6,*) t, y(1) ! y(1): x, pos
22     enddo
23
24     Close(6)
25 End program P3
26
27 ! 4th-order Runge-Kutta subroutine
28 Subroutine rk4(t, dt, y, n)
29     Implicit none
30     real*8, external :: df, dg
31     real*8 :: h, k0, k1, k2, k3, l0, l1, l2, l3
32     real*8, intent(in) :: t, dt
33     real*8, intent(inout), dimension(2) :: y
34     integer, intent(in) :: n
35
36     h=dt/2.0D0
37     k0 = dt * df(y(1),y(n))
38     l0 = dt * dg(y(1),y(n))
39     k1 = dt * df(y(1)+h, y(n)+0.5d0*l0)
40     l1 = dt * dg(y(1)+0.5d0*k0,y(n)+0.5d0*l0)
41     k2 = dt * df(y(1)+0.5d0*k1,y(n)+0.5d0*l1)
42     l2 = dt * dg(y(1)+0.5d0*k1,y(n)+0.5d0*l1)
43     k3 = dt * df(y(1)+k2,y(n)+l2)
44     l3 = dt * dg(y(1)+k2,y(n)+l2)
45     y(1) = y(1) + (k0+2*k1+2*k2+k3)/6.0d0
46     y(n) = y(n) + (l0+2*l1+2*l2+l3)/6.0d0
47     Return
48 End subroutine rk4
49
50 ! function which returns the derivatives (RHS)
51 real*8 function df(y,z)
52 ! dx/dt = v(t)

```

```

53     Implicit none
54     real*8 ,intent(in) :: y, z
55     df=z
56     Return
57 End function df
58
59 real*8 Function dg(y,z)
60 ! dv/dt = -w**2*x(t) -a*v
61 ! the second term is damping term
62     implicit none
63     real*8, intent(in) :: y, z
64     real*8 :: omega, alpha
65     data omega /3.0d0/
66     data alpha /0.5d0/
67
68     dg= - omega**2.0d0 * y - alpha * z
69     Return
70 End function dg

```