Problems Elementary Mechanical Using Python

June 16th 2022

Contents

Listings

Chapter 2

Getting Started with Programming

Problem 2.1 (Seconds) *For this problem we have to*

1. *Write a script that calculates the number of seconds, s, given the number of hours, h, according to the formula $s = 3600\ h$*
2. *Use the script to find the number of seconds in 1.5, 12 and $24\ h$*

Sol.

```
import numpy as np

def seconds(h):
    return 3600*h

if __name__ == '__main__':
    hours = np.array([1.5,12,24])
    for h in hours:
        print(f'{h} hours is equivalent {seconds(h)} seconds')
```

Listing 2.1: Seconds

Problem 2.2 (Spherical mass) *For this problem we have to*

1. *Write a script that calculates the mass of a sphere given its radius r and mass density ρ according to the formula $m = (4\pi/3)\rho r^3$.*
2. *Use the script to find the mass of a sphere of steel of radius $r = 1\ mm$, $r = 1\ m$ and $r = 10\ m$.*

Sol.

```
import numpy as np

def mass(rho,r):
    return (4*np.pi/3)*(rho)*(r**3)

def run():
    rho = 8000
    print(f'The sphere of steel with density {rho} kg/m3')
    radius = np.array([1e-003,1,10])
    for r in radius:
```

```
        print(f'The sphere with {r} m of radius has {mass(rho,r)} kg')

if __name__=='__main__':
    run()
```

Listing 2.2: Spherical mass

Problem 2.3 (Angle) *For this place we have to*

1. *Write a function that for a point (x, y) returns the angle θ from the x-axis using the formula $\theta = \arctan(y/x)$.*
2. *Find the angles θ for the points $(1, 1)$, $(-1, 1)$, $(-1, -1)$, $(1, -1)$.*
3. *How would you change the function to return values of θ in the range $[0, 2\pi]$?*

Sol.

```
import numpy as np

def point(angle):
    return np.arctan(angle[1]/angle[0])

def run():
    angles = np.array([[1,1],[-1,1],[-1,-1],[1,-1]])
    for angle in angles:
        print(f'The point {angle} has angle {point(angle)}')

if __name__=='__main__':
    run()
```

Listing 2.3: Angle

Problem 2.4 (Unit vector) *For this problem we have to*

1. *Write a function that returns the two-dimensional unit vector, (u_x, u_y), corresponding to an angle θ with the x-axis. You can use the formula $(u_x, u_y) = (\cos\theta, \sin\theta)$, where θ is given in radians.*
2. *Find the unit vectors for $\theta = 0, \pi/6, \pi/3, \pi/2, 3\pi/2$.*
3. *Rewrite the function to instead take the argument θ in degrees.*

Sol.

```
import numpy as np

def unit_vector(angle):
    return np.cos(angle), np.sin(angle)

def run():
    angles = [0,np.pi/6,np.pi/3,np.pi/2,3*np.pi/2]
    for angle in angles:
        print(f'The unit vectors for {angle} radiants are {unit_vector(angle)}')

if __name__ == '__main__':
    run()
```

Listing 2.4: Unit vector

Problem 2.5 (Plotting the normal distribution) *The normal distribution, often called the Gaussian distribution, is given as:*

$$P(x;\mu,\sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(x-\mu)^2/(2\sigma)^2} \tag{2.1}$$

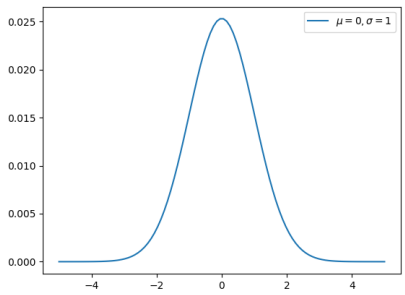
Where mu is the average and sigma is the standard deviation.

1. *Make a function* `normal(x,mu,sigma)` *that returns the normal distribution value* $P(x,\mu,\sigma)$ *as given by the formula 2.1.*
2. *Use this function to plot the normal distribution for* $-5 < x < 5$ *for* $\mu = 0$ *and* $\sigma = 1$.
3. *Plot the normal distribution for* $-5 < x < 5$ *for* $\mu = 0$ *and* $\sigma = 2$ *and* $\sigma = 0.5$ *in the same plot.*
4. *Plot the normal distribution* $-5 < x < 5$ *for* $\sigma = 1$ *and* $\mu = 0, 1, 2$ *in three subplots above each other.*

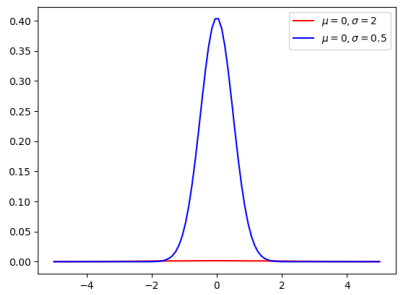
Sol.

```
import numpy as np
import matplotlib.pyplot as plt


def normal(x,mu,sigma):
    return (1/np.square(2*np.pi*(sigma**2)))*np.exp(-((x-mu)**2)/(2*(sigma**2)))

def run_b(x):
    fig, axes =plt.subplots()
    axes.plot(x,normal(x,0,1), label='$\mu=0,\sigma=1$')
    axes.legend()
    fig.savefig('Document/img/chapter2/2-5/2_5_plot_a.png')

def run_c(x):
    fig, axes = plt.subplots(1,1)
    axes.plot(x, normal(x,0,2), 'r', label='$\mu=0,\sigma=2$')
    axes.plot(x, normal(x,0,0.5), 'b', label='$\mu=0,\sigma=0.5$')
    axes.legend()
    fig.savefig('Document/img/chapter2/2-5/2_5_plot_b.png')

def run_d(x):
    fig, (ax1,ax2,ax3) = plt.subplots(3,1)
    ax1.plot(x, normal(x,0,1), 'r', label='$\mu=0,\sigma=1$')
    ax1.legend(loc='upper left')
    ax2.plot(x, normal(x,1,1), 'g', label='$\mu=1,\sigma=1$')
    ax2.legend(loc='upper left')
    ax3.plot(x, normal(x,2,1), 'b', label='$\mu=2,\sigma=1$')
    ax3.legend()
    plt.tight_layout()
    fig.savefig('Document/img/chapter2/2-5/2_5_plot_c.png')

if __name__=='__main__':
    x = np.linspace(-5,5,100)
    run_b(x)
    run_c(x)
    run_d(x)
```

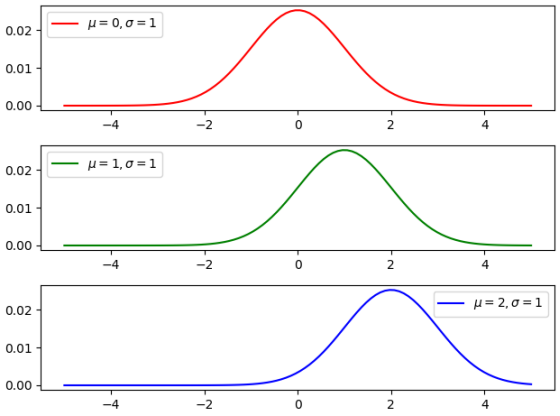
Listing 2.5: Plotting the normal distribution



(a) For $-5 < x < 5,\ \mu = 0$ and $\sigma = 1$



(b) For $-5 < x < 5,\ \mu = 0$ and $\sigma = 2, 0.5$



(c) For $-5 < x < 5,\ \mu = 0, 1, 2$ and $\sigma = 1$

Figure 2.1: Solutions of the problem 2.5

Problem 2.6 (Plotting $1/x^n$) *The function $f(x;n)$ is given as $f(x;n) = x^{-n}$*

1. *Make a function* `f(x,n)` *which returns the value of $f(x;n)$.*
2. *Use this function to plot $1/x$, $1/x^2$ and $1/x^3$ in the same plot for $-1 < x < 1$.*

Sol.

```
import numpy as np
import matplotlib.pyplot as plt

def f(x,n):
    return 1/x**(n)

def run():
    x = np.linspace(-2,2,1000)
    fig, axes = plt.subplots(figsize=(5,5))
    axes.set_title('$1/x^{n}$')
    axes.plot(x,f(x,1),'r', label='$1/x$')
    axes.plot(x,f(x,2),'b', label='$1/x^2$')
    axes.plot(x,f(x,3),'g', label='$1/x^3$')
    axes.set_ylim([-10,10])
```

```
15      axes.set_xlim([-2,2])
16      axes.legend()
17      fig.savefig('Document/img/chapter2/2-6/2_6_plot_xs.png')
18
19  if __name__=='__main__':
20      run()
```

Listing 2.6: Plotting 1/xn

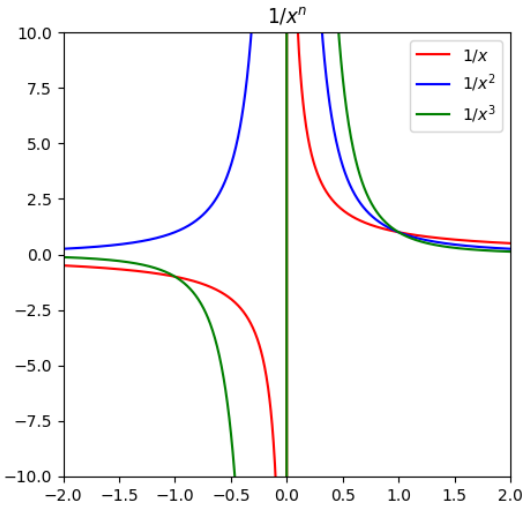


Figure 2.2: Graphs for $n = 1, 2, 3$

Problem 2.7 (Plotting $\sin x/x^n$) *The function $g\{x;n\}$ is given as:*

$$g(x;n) = \frac{\sin x}{x^n} \tag{2.2}$$

1. *Make a function* `gvalue(x,n)` *which returns the value of $g(x;n)$.*
2. *Use this function to plot $\sin x/x$, $\sin x/x^2$ and $\sin x/x^3$ in the same plot for $-5 < x < 5$.*
3. *Use the help function to find out how to place legends for each of the plots into the figure.*

Sol.

```
1  import numpy as np
2  import matplotlib.pyplot as plt
3
4  def g(x,n):
5      return np.sin(x)/x**n
6
7  def run():
8      x = np.linspace(-5,5,500)
9      fig, axes = plt.subplots(figsize=(5,5))
10     axes.set_title('$\sin{x} / x^{n}$')
```

```
    axes.plot(x,g(x,1), label='$\sin{(x)}/x$')
    axes.plot(x,g(x,2), label='$\sin{(x)}/x^{2}$')
    axes.plot(x,g(x,3), label='$\sin{(x)}/x^{3}$')
    axes.legend()
    axes.set_xlim([-5,5])
    axes.set_ylim([-5,5])
    fig.savefig('Document/img/chapter2/2-7/2_7_plot_sinx.png')

if __name__=='__main__':
    run()
```

Listing 2.7: Plotting sin x / xn

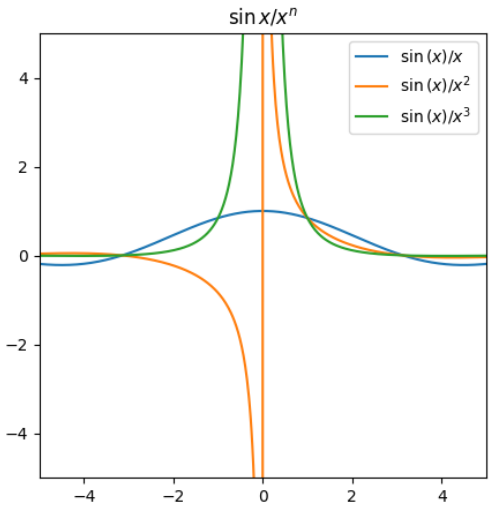


Figure 2.3: Graphs for $n = 1, 2, 3$

Problem 2.8 (Logistic map) *The iterative mapping $x(i+1) = rx(i)(1-x(i))$ is called the logistic map.*

1. *Make a function* `logistic(x,r)` *which returns the value of $x(i+1)$ given $x(i)$ and r as inputs.*
2. *Write a script with a loop to calculate the first 100 steps of the logistic map starting from $x(1) = 0.5$. Store all the values in an array* `x` *with $n = 100$ elements and plot x as a function of the number of steps i:*
3. *Explore the logstic map for $r = 1.0,\ 2.0,\ 3.0$ and 4.0*

Sol.

```
import numpy as np
import matplotlib.pyplot as plt

def logistic(x,r):
    return (r*x)*(1-x)
```

```
6
7  def steps(n,r):
8      x = np.zeros(n)
9      x[0]=0.5 # Star point x(1)=0.5
10
11     for k in range(n-1):
12         x[k+1] = logistic(x[k],r)
13
14     return x
15
16 def run():
17     n=100; i = np.arange(n)
18     x_1 = steps(n,1); x_2 = steps(n,2)
19     x_3 = steps(n,3); x_4 = steps(n,4)
20
21     fig, axes = plt.subplots(figsize=(10,8))
22     axes.set_title('Logistic map')
23     axes.set_xlabel('steps $(i)$')
24     axes.set_ylabel('$x(i)$')
25     axes.plot(i,x_1, label='r=1')
26     axes.plot(i,x_2, label='r=2')
27     axes.plot(i,x_3, label='r=3')
28     axes.plot(i,x_4, label='r=4')
29     axes.legend()
30     fig.savefig('Document/img/chapter2/2-8/2_8_logistic_map.png')
31
32 if __name__=='__main__':
33     run()
```

Listing 2.8: Logistic map

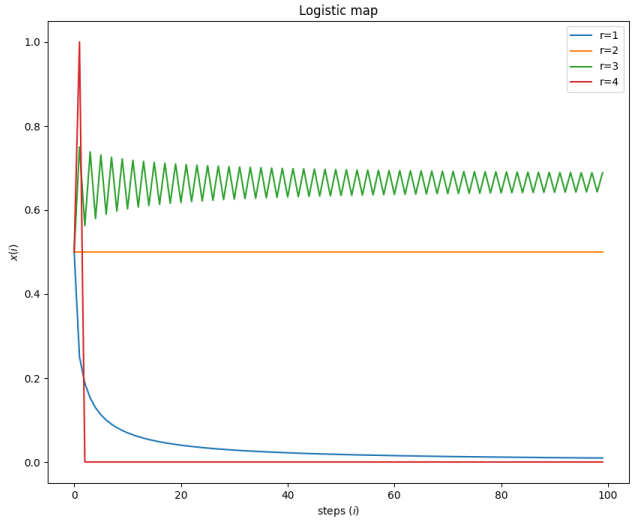


Figure 2.4: Graphs for $r = 1, 2, 3, 4$ and $x(1) = 0.5$

Problem 2.9 (Euler's Method) *In mechanics, we often use Euler's method to determine the motion of an object given how the acceleration depends on the velocity and position of an object. For example, we may know that the acceleration $a(x,v)$ is given as $a(x,v) = -kx - cv$. If we know the position x and the velocity v at a time $t = 0$: $x(0) = x_0 = 0$ and $v(0) = v_0 = 1$, we can use Euler's method to find the position and velocity after a small timestep Δt:*

$$v_i = v(t_{i-1} + \Delta t) = v(t_{i-1}) + a(v(t_{i-1}), x(x_0))\Delta t \tag{2.3}$$

$$x_i = x(t_{i-1} + \Delta t) = x(t_{i-1}) + v(t_{i-1})\Delta t \tag{2.4}$$

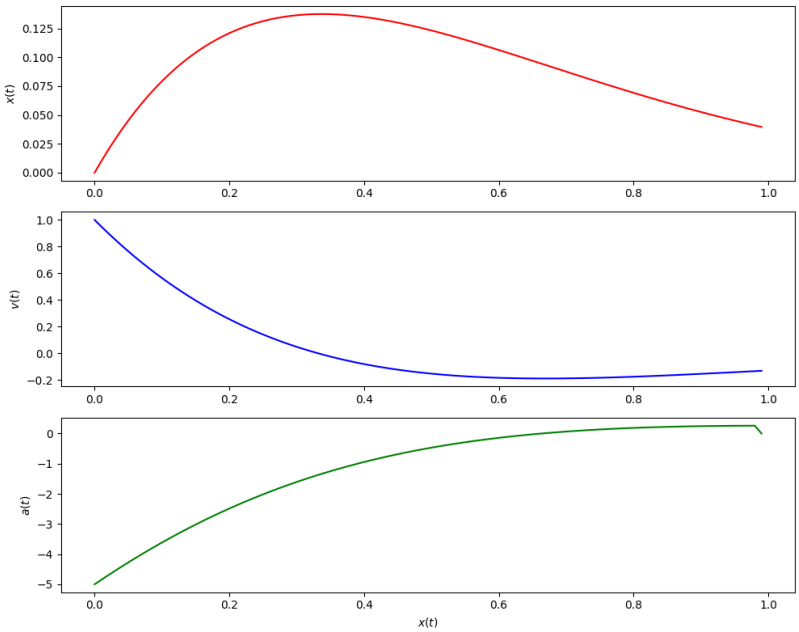
and so on. We can therefore use this scheme to find the position $x(t)$ and the velocity $v(t)$ as function of time at the discrete values $t_i = i\Delta t$ in time.

1. *Write a function* `acceleration(v,x,k,C)` *which returns the value of $a(x,v) = -kx - Cv$.*
2. *Write a script that calculates the first 100 values of $x(t_i)$ and $v(t_i)$ when $k = 10$, $C = 5$ and $\Delta t = 0.01$. Plot $x(t)$, $v(t)$ and $a(t)$ as functions of time.*
3. *What would you need to change to instead find $x(t)$ and $v(t)$ is the acceleration was given as $a(v,x) = k\sin x - Cv$?*

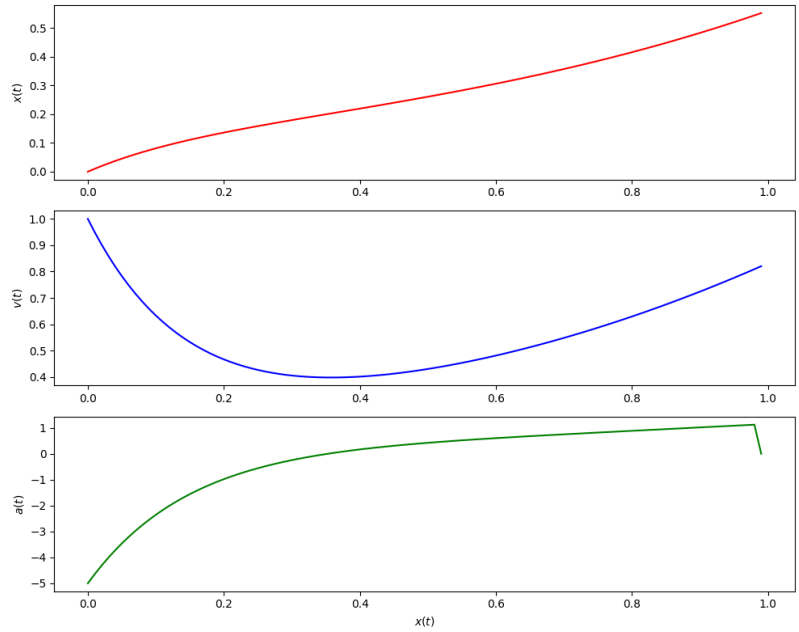
Sol.

```
import numpy as np
import matplotlib.pyplot as plt

def acceleration(v,x,k,C):
    return -k*x - C*v

def acceleration_2(v,x,k,C): # Part 3
    return k*np.sin(x) - C*v

def plot(function,name):
    k=10; C=5; delta=0.01; n=100
    x = np.zeros(n); v = np.zeros(n)
    a = np.zeros(n); t = np.zeros(n)
    x[0]=0; v[0]=1

    for i in range(n-1):
        a[i] = function(v[i],x[i],k,C)
        x[i+1] = x[i] + v[i]*delta
        v[i+1] = v[i] + a[i]*delta
        t[i+1] = t[i] + delta

    fig, (ax1,ax2,ax3) = plt.subplots(3,1, figsize=(10,8))
    ax1.plot(t,x, 'r')
    ax1.set_ylabel('$x(t)$')
    ax2.plot(t,v, 'b')
    ax2.set_ylabel('$v(t)$')
    ax3.plot(t,a, 'g')
    ax3.set_ylabel('$a(t)$')
    ax3.set_xlabel('$x(t)$')
    plt.tight_layout()
    fig.savefig('Document/img/chapter2/2-9/' + name)

if __name__=='__main__':
    plot(acceleration,'2_9_accele_a.png')
    plot(acceleration_2,'2_9_accele_b.png')
```

Listing 2.9: Euler's method



(a) Graph for $a(x, v) = -kx - Cv$



(b) Graph for $a(x, v) = k \sin x - Cv$

Figure 2.5: Solutions of the problem 2.9

Problem 2.10 (Throwing two dice) *You throw a pair of six-sided dice and sum the number from each of the dice: $Z = X_1 + X_2$, where Z is the sum of the results from dice 1, X_1, and dice 2, X_2. If we perform this experiment many times (N), we can find the average and standard deviation from standard estimators from statistics. The average, $\langle Z \rangle$, of Z is estimated from:*

$$\langle Z \rangle = \frac{1}{N} \sum_{j=1}^{N} Z_j \tag{2.5}$$

and the standard deviation, ΔZ, is estimated from:

$$\Delta Z = \frac{1}{N-1} \left(\sum_{j=1}^{N} (Z_j - \langle Z \rangle) \right)^2 \tag{2.6}$$

1. *Write a function that returns an array of N values for Z.*
2. *Write a function that returns an estimate of the average of an array* `z` *using the formula provided.*
3. *Write a function that returns an estimate of the standard deviation of an array* `z` *using the formula provided.*
4. *Find the average and standard deviation for $N = 100$ throws of two dice.*

Sol.

```python
import numpy as np

def dice(n):
    x_1 = np.random.randint(1,6,n)
    x_2 = np.random.randint(1,6,n)
    return x_1 + x_2

def average(z,n):
    return (1/n)*sum(z)

def standard_deviation(z,Z):
    i=0; DZ=0
    while i!=n:
        DZ+=(z[i]-Z)**2
        i+=1

    return (1/(n-1))*DZ

if __name__=='__main__':
    n=100
    z = dice(n)
    Z = average(z,n)
    x = standard_deviation(z,Z)
    print(f'The average for {n} times is {Z} and the standard deviation is {x}')
```

Listing 2.10: Throwing two dice

Problem 2.11 (Reading data) *The file trajectory.dat*[1] *contains a list of numbers*

[1] https://folk.universitetetioslo.no/malthe/mechbook/trajectory.dat

```
t0  x0  y0
t1  x1  y1
..  ..  ..
tn  xn  yn
```

corresponding to the time `t(i)` *measured in seconds, and the* x *and* y *positions* `x(i)` *and* `y(i)` *measured in meters for the trajectory of a particle.*

1. *Read the data file into the arrays* `t,x` *and* `y`.
2. *Plot the* x *and* y *positions as function of time in two plots above each other.*
3. *Plot the* (x, y) *position of the object in a plot with* x *and* y *on the two axes.*

Sol.

```
1
2  import numpy as np
3  import matplotlib.pyplot as plt
4
5  def select_data(data):
6      t = data[:,0]; x = data[:,1]
7      y = data[:,2]
8      return t, x, y
9
10 def w_vs_t(t,x,y):
11     fig, (ax1,ax2) = plt.subplots(2,1,figsize=(6,6))
12     ax1.set_title('$t$ vs $x(t)$, $y(t)$')
13     ax1.plot(t,x,'r')
14     ax1.set_ylabel('$x(t)$')
15     ax1.set_xlabel('$t$')
16     ax2.plot(t,y,'b')
17     ax2.set_ylabel('$y(t)$')
18     ax2.set_xlabel('$t(s)$')
19     fig.savefig('Document/img/chapter2/2-11/2_11_b.png')
20
21 def x_y_position(x,y):
22     fig, axes = plt.subplots(1,1)
23     axes.plot(x,y,'g')
24     axes.set_title('$(x,y)$')
25     axes.set_xlabel('$x$')
26     axes.set_ylabel('$y$')
27     fig.savefig('Document/img/chapter2/2-11/2_11_c.png')
28
29 if __name__=='__main__':
30     data = np.loadtxt('Document/datasets/chapter2/trajectory.dat')
31     t, x, y = select_data(data)
32     w_vs_t(t,x,y)
33     x_y_position(x,y)
```

Listing 2.11: Reading data

Problem 2.12 (Numerical integration of a data-set) *The file velocityy.dat*[2] *contains a list of numbers*

[2] https://folk.universitetetioslo.no/malthe/mechbook/velocityy.dat

```
t0  v0
t1  v1
..  ..
tn  vn
```

correspondingto the time `t(i)` *measured in seconds, and the velocity* `v(i)` *in meters per second for the trajectory of a projectile.*

1. *Read the data file into the arrays* `t` *and* `v`.
2. *Plot* $v(t)$ *as function of time.*
   beginalign For a data-set* `t(i), v(i)`*, you can estimate the function corresponding to the integral of* $v(t)$ *with respect to* t *at the times* t_i *using the iterative scheme.*

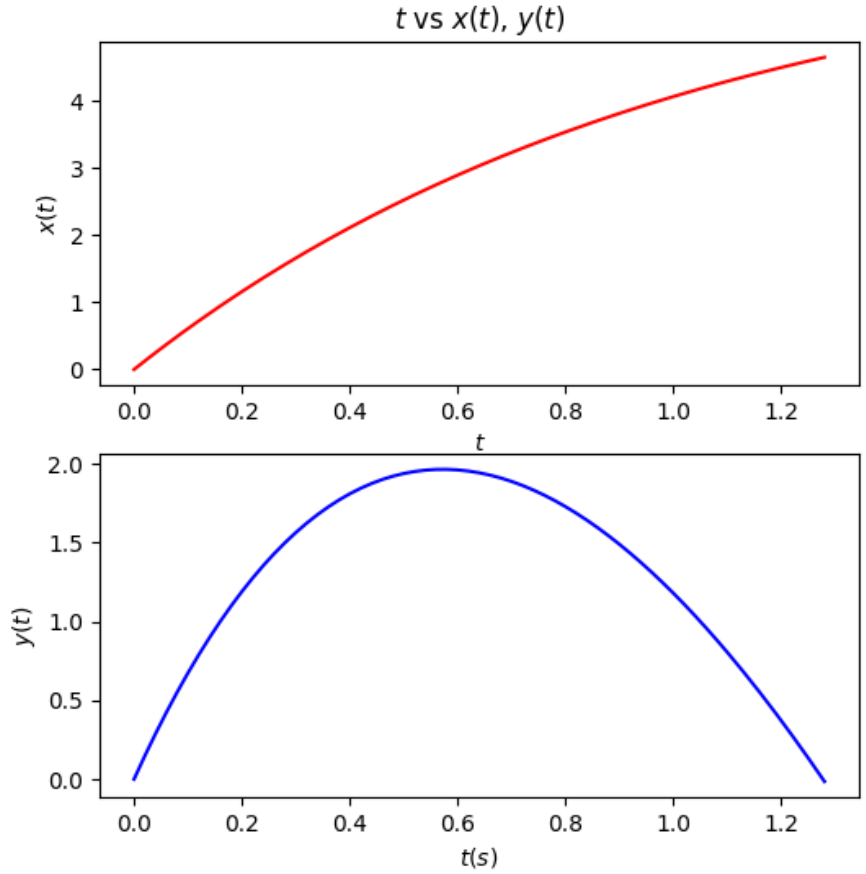
$$y(t_n) \approx y(t_{n-1}) + v(t_{n-1})(t_n - t_{n-1}) \tag{2.7}$$

   where $v(t_i)$=`v(i)` *and* t_i=`t(i)`*. You can assume that the motion starts at* $y(t_0) = 0.0m$ *at* $t = t_0$.
3. *Write a script to calculate the time integral* $y(t_i)$ *of the dataset using this formula implement using a* `for`*-loop.*
4. *Plot the position* $y(t)$ *and the derivative* $v(t)$ *as functions of time in two plots above each other.*

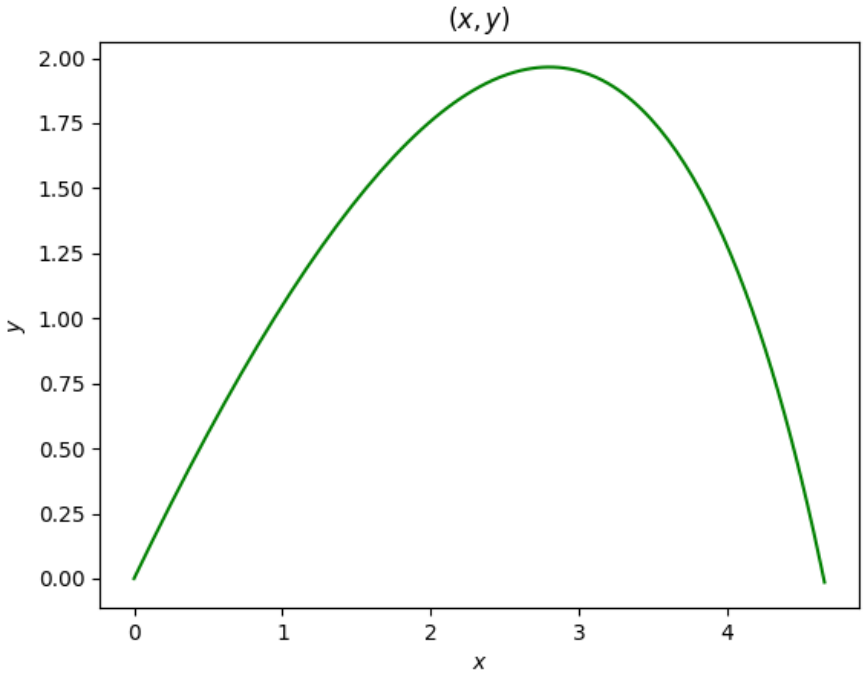
Sol.

```python
import numpy as np
import matplotlib.pyplot as plt

def reading_txt(string):
    data = np.loadtxt(string)
    t = data[:,0]; v = data[:,1]
    return t, v

def location(t,v):
    y = np.zeros(len(t))

    for k in range(1,len(t)):
        y[k] = y[k-1] + v[k-1]*(t[k]-t[k-1])

    return y

def plotting(t,y,v):
    fig, (ax1,ax2) = plt.subplots(2,1,figsize=(6,5))
    ax1.plot(t,y,'r')
    ax1.set_ylabel('$y(t)$')
    ax2.plot(t,v,'b')
    ax2.set_ylabel('$v(t)$')
    ax2.set_xlabel('$t(s)$')
    plt.tight_layout()
    fig.savefig('Document/img/chapter2/2-12/2_12-png')

if __name__ == '__main__':
    t, v = reading_txt('Document/datasets/chapter2/velocity.dat')
    y = location(t,v)
    plotting(t,y,v)
```

Listing 2.12: Numerical integration



(a) Graph x and y as time functions



(b) Graph (x, y) position

Figure 2.6: Solutions of the problem 2.11

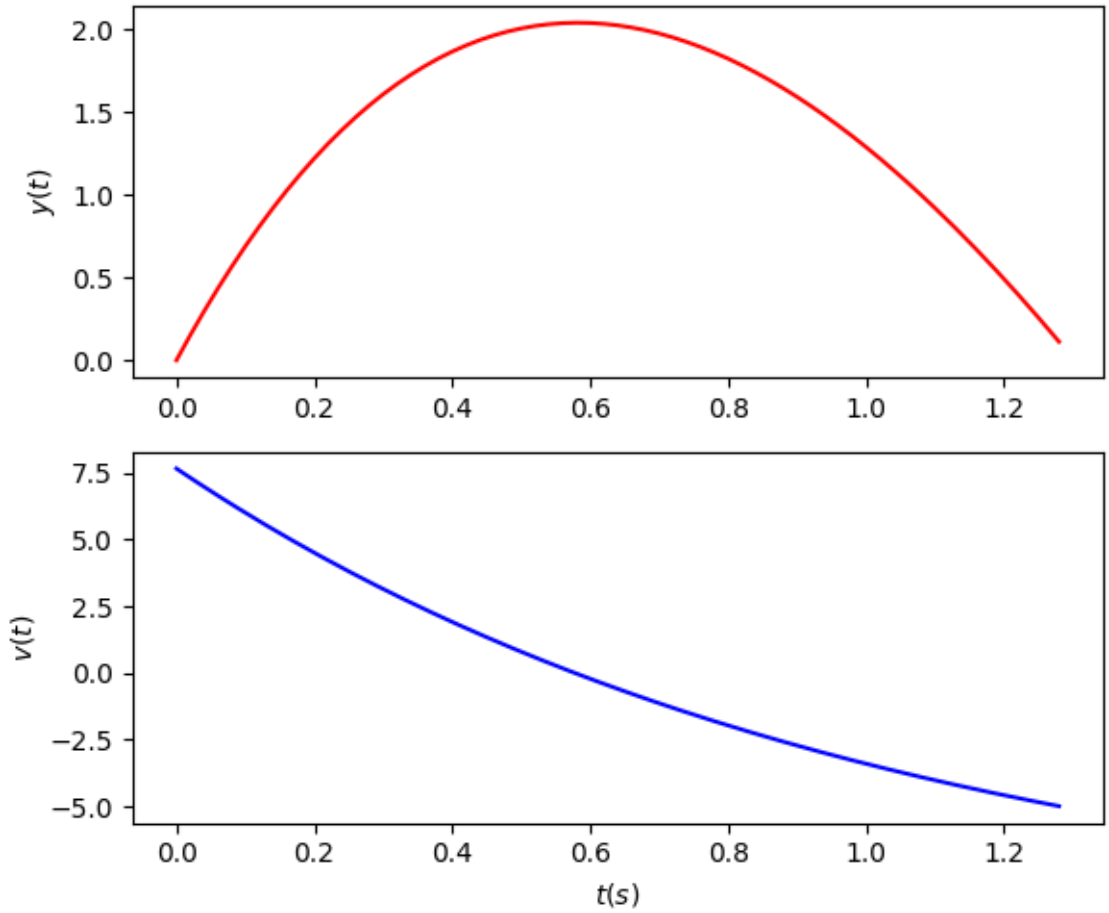


Figure 2.7: Position $y(t)$ and derivative $v(t)$, solution of the problem 2.12

Chapter 3

Units and Measurement

Problem 3.1 (Kilometers per hour) *A car os driving at 144 km/h. Find the velocity in m/s*

Problem 3.2 (Miles per hour) *You car speedometer is showing both km/h and mph, miles per hour. 1 miles is 1609.34 m.*

1. *If you speedometer is showing 70 km/h. What does it show in mph?*
2. *If you speedometer is showing 55 mph. What does it show in km/h*

Problem 3.3 (Acceleration of gravity) *The acceleration of gravity is approximately* $g = 9.8\ m/s^2$.

1. *Find the acceleration of gravity in feet per second squared,* ft/s^2. *1 foot is 0.3048 m.*
2. *Find the acceleration of gravity in kilometers per hour squared,* km/h^2.

Problem 3.4 (Bacterial volume) *A bacteria is like a cylinder with legnth* $4\mu m$ *and radius* $1\ \mu m$.

1. *Find its volume in* μm^3.
2. *Find its volume in* m^3.
3. *Find its volume in liters.*

Problem 3.5 (Ruler length) *You have a platinum-iridium ruler that you have measured to be 0.11236 m using a high precision method. You use the ruler to measure the length of your desk, and find that the ruler fits about 20 times across the desk. What is the length of your desk?*

Problem 3.6 (Sphere mass and volume) *A small steel sphere has a radius of 1.2 mm.*

1. *What is the volume of the sphere?*
2. *The density of the particular steel alloy used is* $\rho = 7782kg/m^3$. *What is the mass of the sphere?*

Problem 3.7 (Laserlength) *You use a laser distance measurer to measure the distance from one wall to another in your house. It reads 11.2 m. As you walk across to the other wall, you see that there is a small protrusion from the wall. Using your tape measure, you find that the protrusion is 5 mm high. What is the distance from the other wall to the protrusion?*

Problem 3.8 (Salmon speed) *You have designed a special circuit to measure the swimming speed of a salmon. The circuit has a length of 62.8 m. You measure the time a salmon takes to swim one lap to be 20.6 s.*

1. *What is the swimming speed of a salmon?*
   Your assistant insists that you would get better precision if you instead measured the time the salmon took to swim 10 rounds. You find that the salmon uses 206.0 s to swim 10 rounds.
2. *What is the speed of the salmon?*
3. *Does this produce better accuracy? Can you give other examples of situations where this strategy would improve the accuracy?*