Problems Elementary Mechanical Using Python

June 16th 2022

Chapter 2

First time

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.

```python
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')
```

```
1.5 hours is equivalent 5400.0 seconds
12.0 hours is equivalent 43200.0 seconds
24.0 hours is equivalent 86400.0 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.

```python
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()
```

```
The sphere of steel with density 8000 kg/m3
The sphere with 0.001 m of radius has 3.351032163829113e-05 kg
The sphere with 1.0 m of radius has 33510.32163829113 kg
The sphere with 10.0 m of radius has 33510321.638291128 kg
```

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 $[0, 2\pi]$?*

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()
```

```
The unit vectors for 0 radiants are (1.0, 0.0)
The unit vectors for 0.5235987755982988 radiants are (0.86602540378, 0.49999999999)
The unit vectors for 1.0471975511965976 radiants are (0.50000000000, 0.86602540378)
The unit vectors for 1.5707963267948966 radiants are (6.123233995736766e-17, 1.0)
The unit vectors for 4.71238898038469 radiants are (-1.8369701987210297e-16, -1.0)
```

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

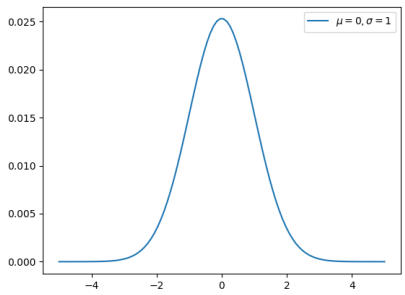
$$P(x;\mu,\sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(x-\mu)^2/(2\sigma)^2} \tag{2.1}$$

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

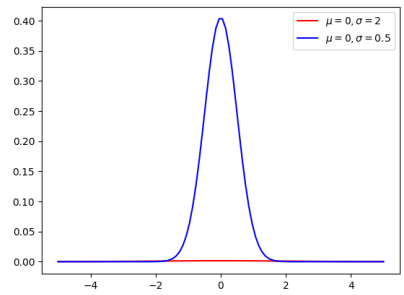
1. *Make a function* `normal(x,mu,sigma)` *that returns the normal distribution value* $P(x,\mu,\sigma)$ *as given by the formula 2.1.*
2. *Use this function to plot the normal distribution for* $-5 < x < 5$ *for* $\mu = 0$ *and* $\sigma = 1$.
3. *Plot the normal distribution for* $-5 < x < 5$ *for*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

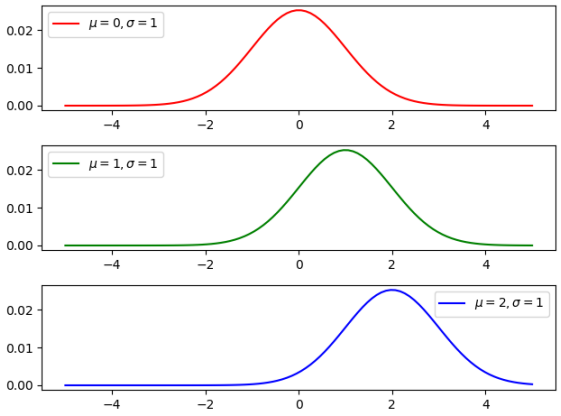
```
import numpy as np
import matplotlib.pyplot as plt


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)
```



(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])
```

```
    axes.set_xlim([-2,2])
    axes.legend()
    fig.savefig('Document/img/chapter2/2-6/2_6_plot_xs.png')

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

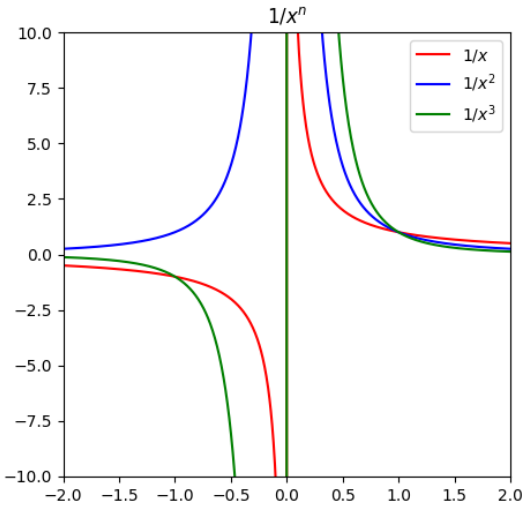


Figure 2.2: Graphs for $n = 1, 2, 3$

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.

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

def g(x,n):
    return np.sin(x)/x**n

def run():
    x = np.linspace(-5,5,500)
    fig, axes = plt.subplots(figsize=(5,5))
    axes.set_title('$\sin{x} / x^{n}$')
    axes.plot(x,g(x,1), label='$\sin{(x)}/x$')
    axes.plot(x,g(x,2), label='$\sin{(x)}/x^{2}$')
```

```
13      axes.plot(x,g(x,3), label='$\sin{(x)}/x^{3}$')
14      axes.legend()
15      axes.set_xlim([-5,5])
16      axes.set_ylim([-5,5])
17      fig.savefig('Document/img/chapter2/2-7/2_7_plot_sinx.png')
18
19  if __name__=='__main__':
20      run()
```

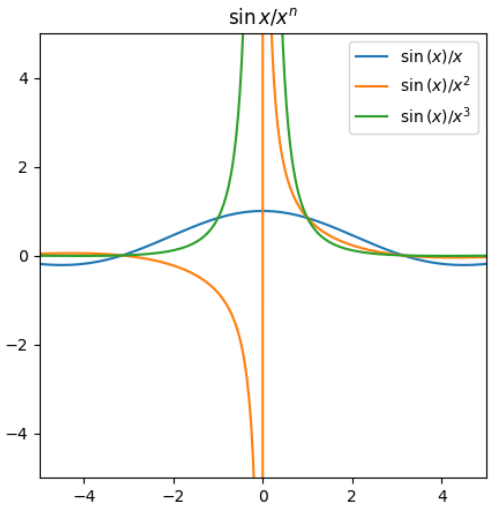


Figure 2.3: Graphs for $n = 1, 2, 3$

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.

```
1  import numpy as np
2  import matplotlib.pyplot as plt
3
4  def logistic(x,r):
5      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()
```

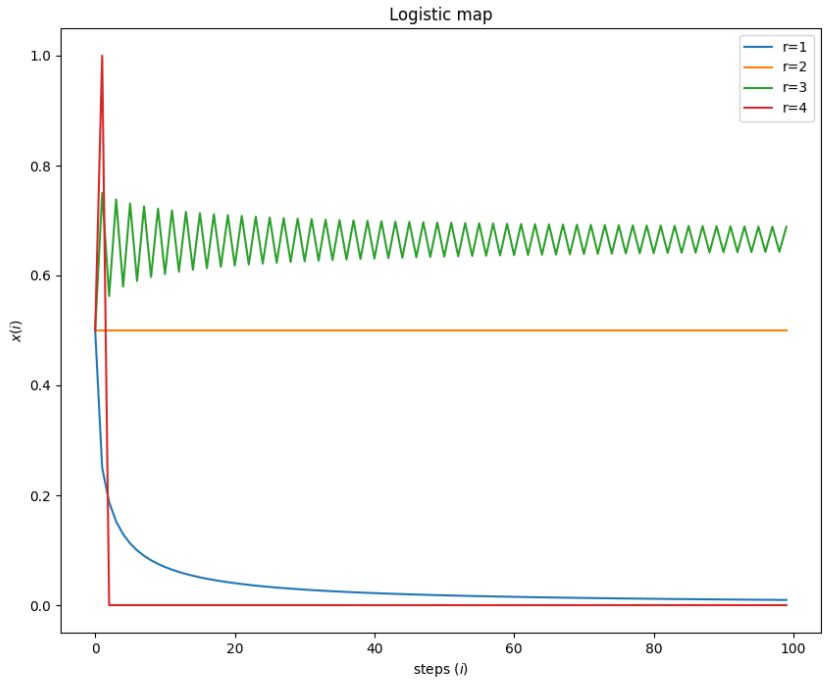


Figure 2.4: Graphs for $r = 1, 2, 3, 4$ and $x(1) = 0.5$

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:*

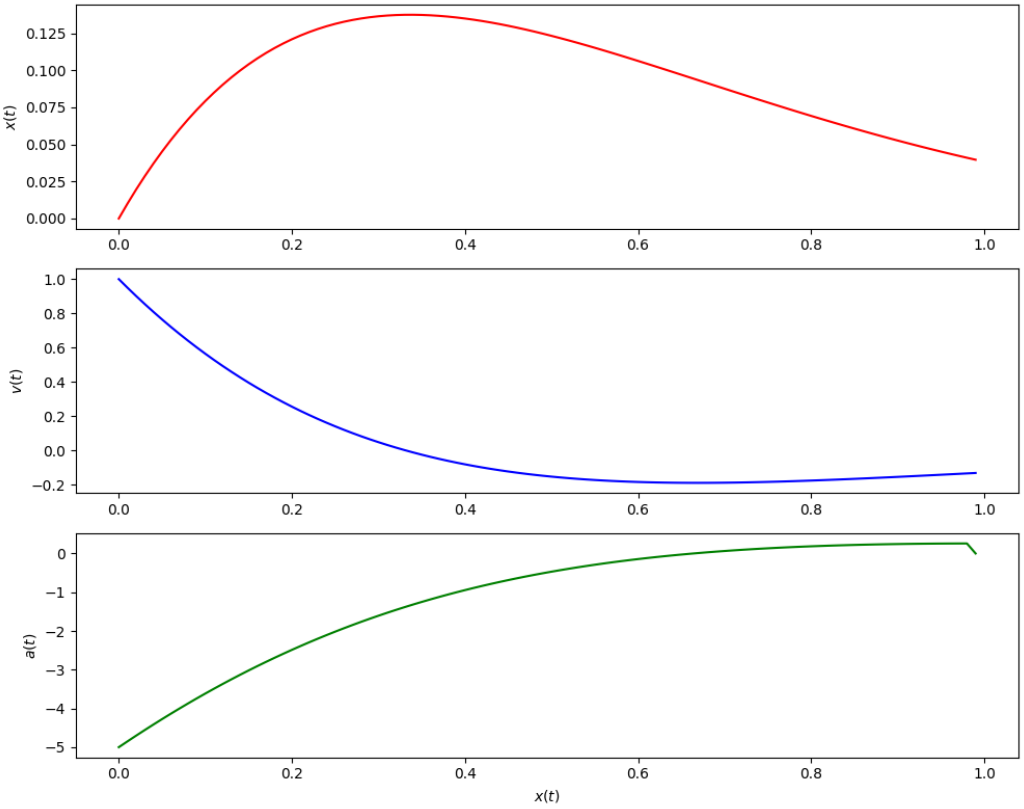
$$v_i = v(t_{i-1} + \Delta t) = v(t_{i-1}) + a(v(t_{i-1}), x(x_0))\Delta t \tag{2.3}$$
$$x_i = x(t_{i-1} + \Delta t) = x(t_{i-1}) + v(t_{i-1})\Delta t \tag{2.4}$$

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.

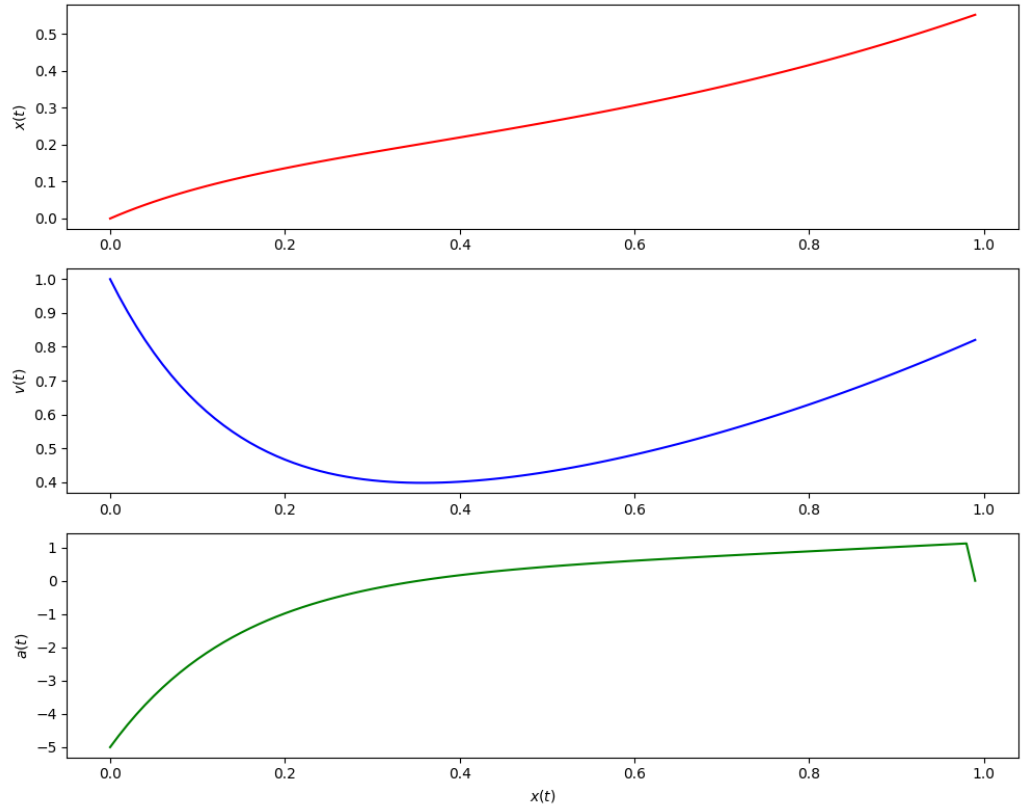
1. *Write a function* `acceleration(v,x,k,C)` *which returns the value of $a(x,v) = -kx - Cv$.*
2. *Write a script that calculates the first 100 values of $x(t_i)$ and $v(t_i)$ when $k = 10$, $C = 5$ and $\Delta t = 0.01$. Plot $x(t)$, $v(t)$ and $a(t)$ as functions of time.*
3. *What would you need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

Sol.

```python
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')
```



(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