Computational Physics I

Name: Sudip KC

Physics | Class of 2022 | Jacobs University Bremen

Falling body problem:

In the problem, a body is falling under a velocity-dependent damping due to the damping force F_D as well as the height-dependent gravitational field of the earth due to gravitational force F_G. Mathematically:

$$F_G - F_D = ma$$

$$a = \frac{-G \cdot M}{R^2 \cdot \left(1 + \frac{y}{R}\right)^2} + \frac{k}{m} \cdot v^2$$

The terminal velocity can be calculated using the formula:

$$v_{terminal} = \sqrt{\frac{(m \cdot g)}{k}}$$

For our given height of 5000 m, the terminal velocity was not acheived.

The respective graph of position, velocity, and acceleration vs time were observed respectively. Obtained reasult was not as expected for velocity as the object never reached the terminal velocity because of which acceleration never reached zero. The same computation can be calculated for higher heights and bigger k value as shown below in the graph where the velocity vs time graph reaches the terminal velocity.

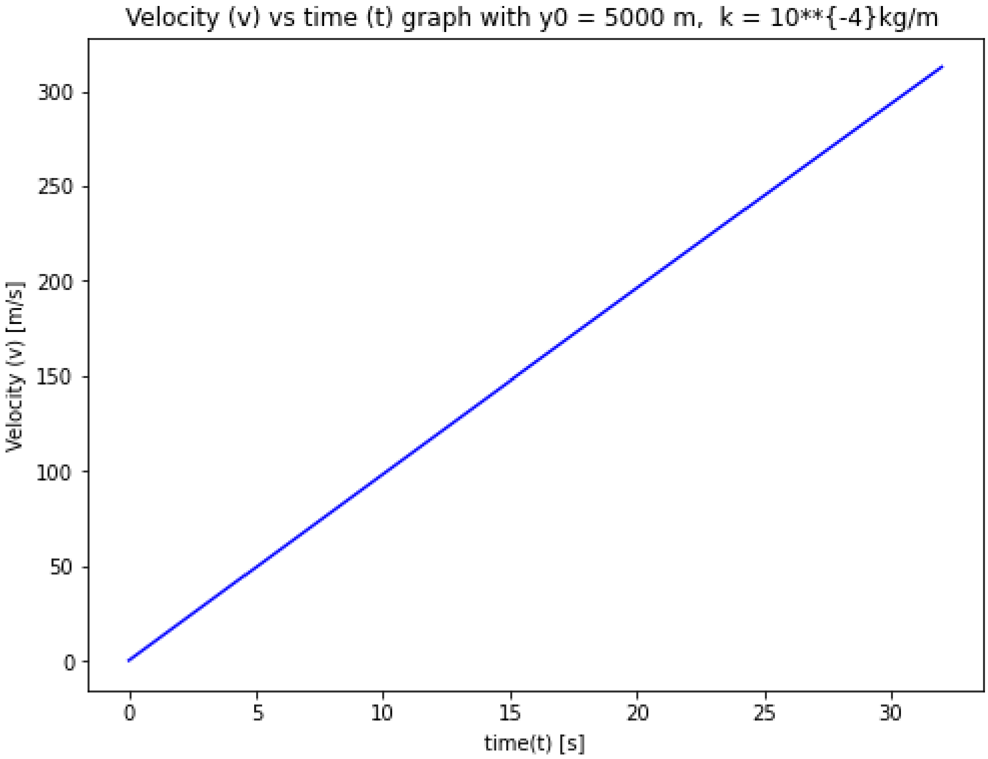
Validation of Euler Algoroithm

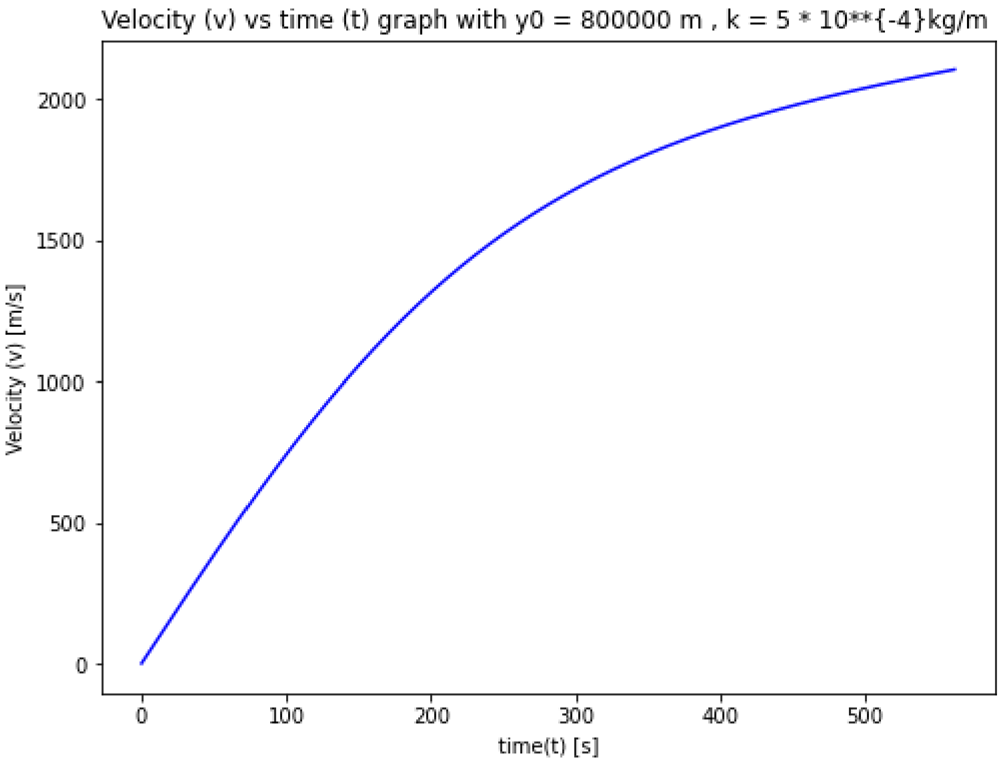
For the validation of the computed euler algorithm a special case with k=0 i.e. no velocity based damping was observed. In the case we have analytical solution to the problem which is shown below as postion vs time graph.

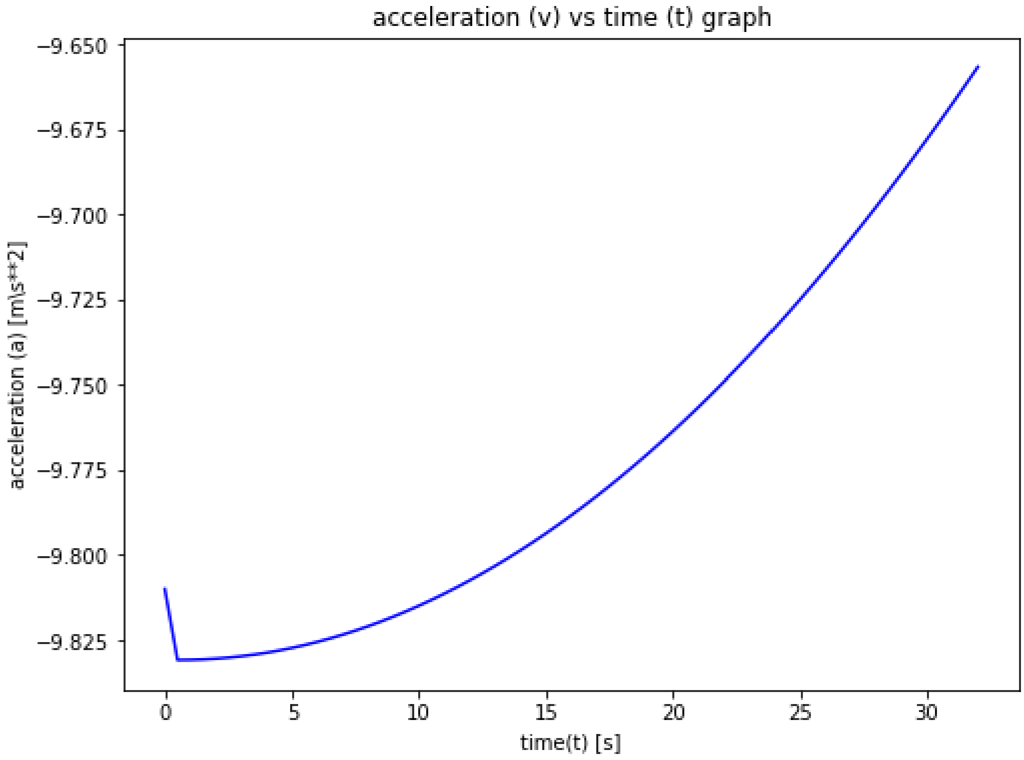
In [4]:

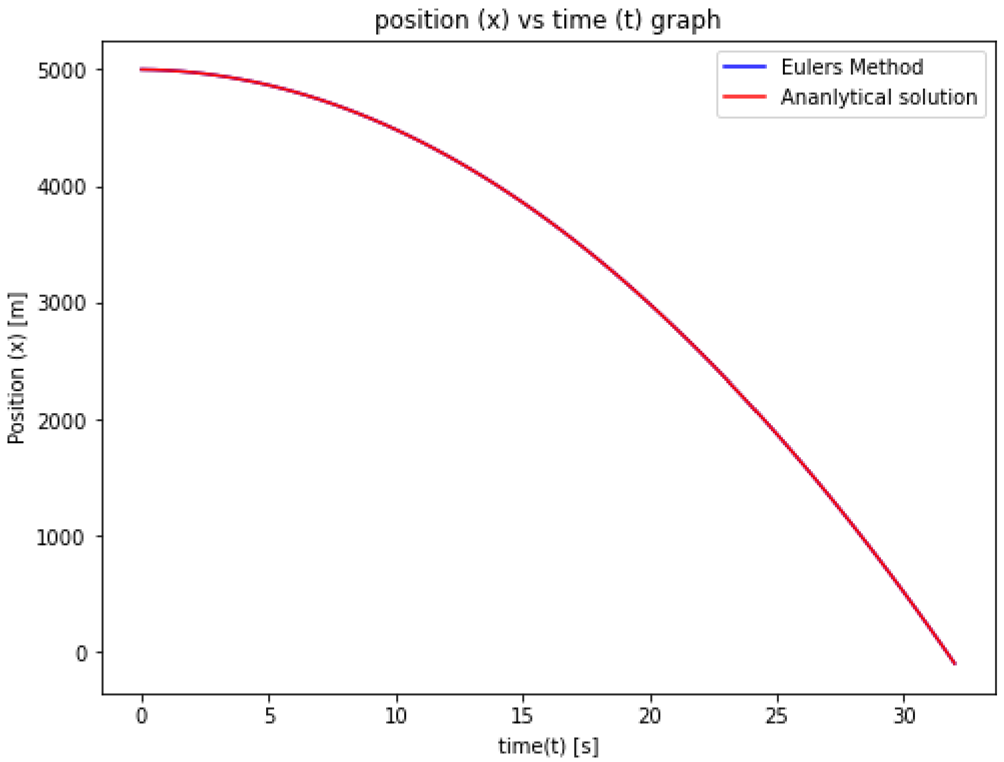
```
import numpy as np
import matplotlib.pyplot as plt

#Setting the constants and initial conditions

M = 5.99 *10**(24)              # Mass of the earth
m = 50                           # mass of the object
k = 10**(-4)
G = 6.67*10**(-11)                  # Gravitational constant
R = 6370000                    # Radius of the earth
t0 = 0
y0 = 5000
v0 = 0
g = -9.81                           # Accelertion due to gravity

#velocity-dependent daming as well as the height-dependent acceleration
```

```python
def acc(y, v):
    return -(G*M)/(R**2*(1 + y/R)**2) + (k/m)*v**2

#Euler method
def Euler(y0, k):

    (y, v, t)=(y0, v0, t0)
    yarr, varr,tarr, aarr = [y0], [v0], [t0], [g]

    dt = 0.5
    while y>0:
        a = acc(y, v)
        v = v - dt*a
        y = y - dt*v
        t = t + dt
        tarr.append(t)
        yarr.append(y)
        varr.append(v)
        aarr.append(a)
    return [tarr, yarr, varr, aarr]

t1, y1, v1, a1 = Euler(5000, 10**(-4))
t1, y1a, v1, a1 = Euler(5000, 0)
t2, y2, v2, a2 = Euler(800000, 5*10**(-4))


def plot(x, y, xlabel, ylabel, title):
    plt.figure(1, figsize=(8,6))
    plt.plot(x, y, 'b-')
    plt.xlabel(xlabel)
    plt.ylabel(ylabel)
    plt.title(title)
    plt.show()

plot (t1, v1, 'time(t) [s]','Velocity (v) [m/s]', 'Velocity (v) vs time (t) g
plot (t2, v2, 'time(t) [s]','Velocity (v) [m/s]', 'Velocity (v) vs time (t) g
plot (t1, a1, 'time(t) [s]','acceleration (a) [m\s**2]', 'acceleration (v) vs

#plotting x(t)
plt.figure(1, figsize=(8,6))
plt.plot(t1, y1, 'b-', label = 'Eulers Method')
plt.plot(t1, y1a, 'r-', label = 'Ananlytical solution')
plt.xlabel('time(t) [s]')
plt.ylabel('Position (x) [m]')
plt.title('position (x) vs time (t) graph')
plt.legend()
plt.show()
```


Velocity (v) vs time (t) graph with y0 = 5000 m, k = 10**{-4}kg/m
Velocity (v) [m/s]
time(t) [s]
0
50
100
150
200
250
300
0
5
10
15
20
25
30


Velocity (v) vs time (t) graph with y0 = 800000 m , k = 5 * 10**{-4}kg/m
Velocity (v) [m/s]
time(t) [s]
0
500
1000
1500
2000
0
100
200
300
400
500





In [ ]:

Simple Harmonic Oscillator with Eulers Method

The code below describes the equation of motion of a simple harmonic oscillator for which:

$$F = -k \cdot x$$

Acceleration can then be defined as:

$$a = \frac{-k \cdot x}{m}$$

Analytically the postion of the particle in SHM was calculated using the equation:

$$y = y_0 \sin \omega t \quad \omega = \sqrt{\frac{k}{m}}$$

Also, the positon was calulated using Eulers method and the observed reasult is plotted below as a function of position vs time graph. As the plotted analytical and numerical values were not exactly overlapping, the new Euler Richardson method was also used. However, the position time graph using Euler Richardson method was even worse compared to the one with Eulers method.

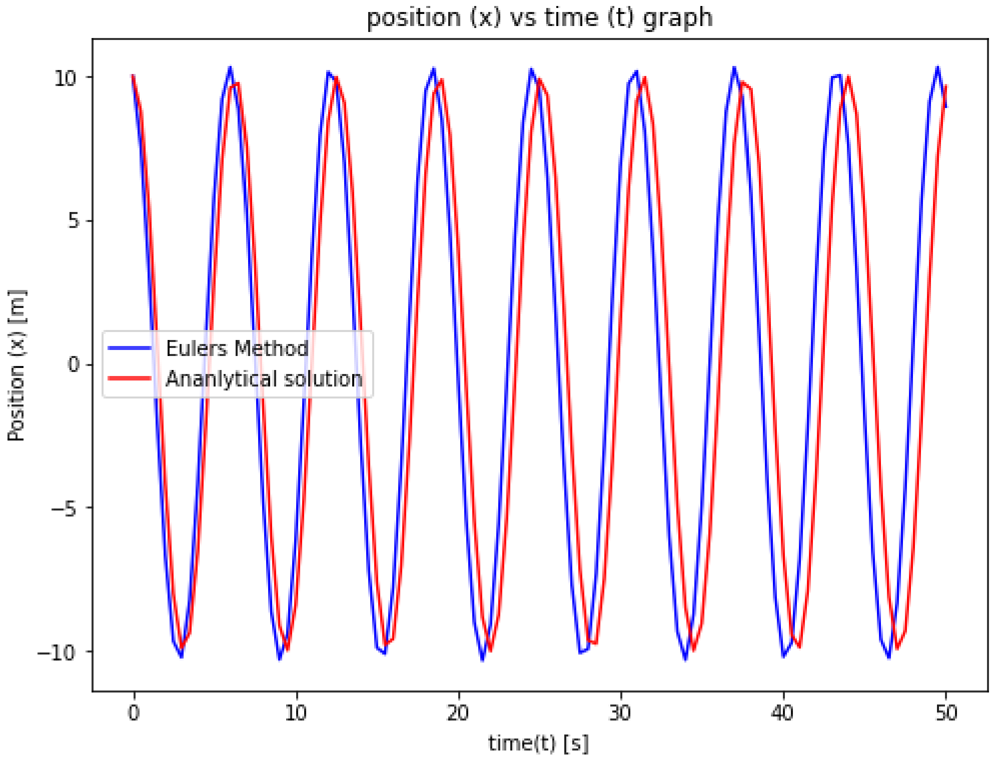
Checks performed

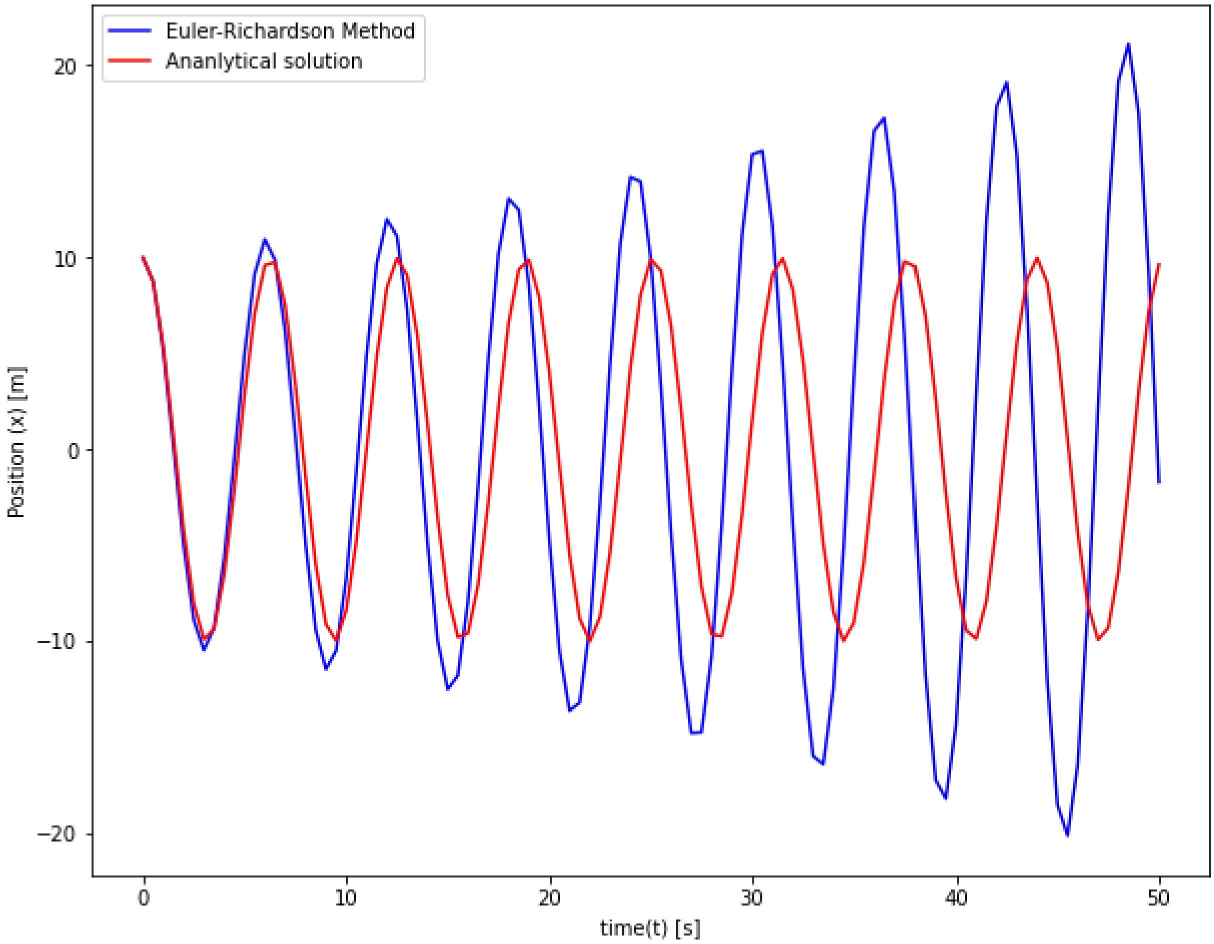
In order to validate the code, a case with k = 0 was taken with certain value of the velocity where position was linearly increasing with time as expected.

In [97]:

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

#Setting the constants and initial conditions


m = 1
k = 1
t0 = 0
y0 = 10
v0 = 0
g = -9.81

# Defining accelaration for a object in harmonic oscillator

def acc(y, v):
    return -(k/m)*y

#Euler method

(y, v, t)=(y0, v0, t0)
yarr, varr,tarr, aarr = [y0], [v0], [t0], [g]

dt = 0.5
while t<50:
    a = acc(y, v)
    v = v + dt*a
    y = y + dt*v
    t = t + dt
    tarr.append(t)
    yarr.append(y)
    varr.append(v)
    aarr.append(a)




#  Numerical error in the position of SHM in Eulers method.

pos, err  = [], []
for i in range(len(tarr)):
    ypos = y0 * np.cos(tarr[i])
```

```
    e = np.absolute(ypos - yarr[i])
    pos.append(ypos)
    err.append(e)

#plotting x(t)
plt.figure(1, figsize=(8,6))
plt.plot(tarr, yarr, 'b-', label = 'Eulers Method')
plt.plot(tarr, pos, 'r-', label = 'Ananlytical solution')
plt.xlabel('time(t) [s]')
plt.ylabel('Position (x) [m]')
plt.title('position (x) vs time (t) graph')
plt.legend()
plt.show()



#Euler-Richardson Scheme

dt = 0.5
(y, v, t)=(y0, v0, t0)
yarr, varr,tarr, aarr = [y0], [v0], [t0], [g]
while t<50:
    a = acc(y, v)
    vmid = v + 0.5*a*dt
    ymid = y + 0.5*v*dt
    a = acc(ymid, vmid)
    v = v + a*dt
    y = y + vmid*dt
    t=t+dt
    tarr.append(t)
    yarr.append(y)
    varr.append(v)
    aarr.append(a)



# Calculting Analytical solution of Harmonic Oscillator

pos, err  = [], []
for i in range(len(tarr)):
    ypos = y0 * np.cos(tarr[i])
    e = np.absolute(ypos - yarr[i])
    pos.append(ypos)
    err.append(e)

#plotting x(t)
plt.figure(1, figsize=(10,8))
plt.plot(tarr, yarr, 'b-', label = 'Euler-Richardson Method')
plt.plot(tarr, pos, 'r-', label = 'Ananlytical solution')
plt.xlabel('time(t) [s]')
plt.ylabel('Position (x) [m]')
plt.legend()
plt.show()
```


position (x) vs time (t) graph
Eulers Method
Ananlytical solution
Position (x) [m]
time(t) [s]
10
5
0
−5
−10
0
10
20
30
40
50


Euler-Richardson Method
Ananlytical solution
Position (x) [m]
time(t) [s]
20
10
0
−10
−20
0
10
20
30
40
50

In [ ]:

In [ ]:

In [ ]: