Computational Physics
Fall 2020

Lalit Chaudhary

September 25, 2020

Project 1

Contents

List of Figures

1 Euler's Algorithm

```
#Euler method
y, v, t = y0, v0, t0
yarr, varr, tarr = [y], [v], [t]
while y > 0:

    a = ac(y, v)
    v = v - a*dt
    y = y - v*dt
    t = t+dt
    tarr.append(t)
    varr.append(v)
    yarr.append(y)
```

Figure 1: Code Snippet for Euler's Algorithm

2 Validation of Euler Algorithm

The following checks were performed to validate the code for section 1 using Euler algorithm.

2.1 Special Case of No Damping

In case of $k = 0$ i.e. no velocity based damping, we have a analytical solution to the problem. Figure(2) shows the graph obtained for this special case.

Here, the slight deviation of the algorithm and analytical solution is due to the fact that the acceleration due to gravity ($'g'$) is calculated for each time interval in case of Euler's Method while we assume the acceleration is constant in case of analytical solution.

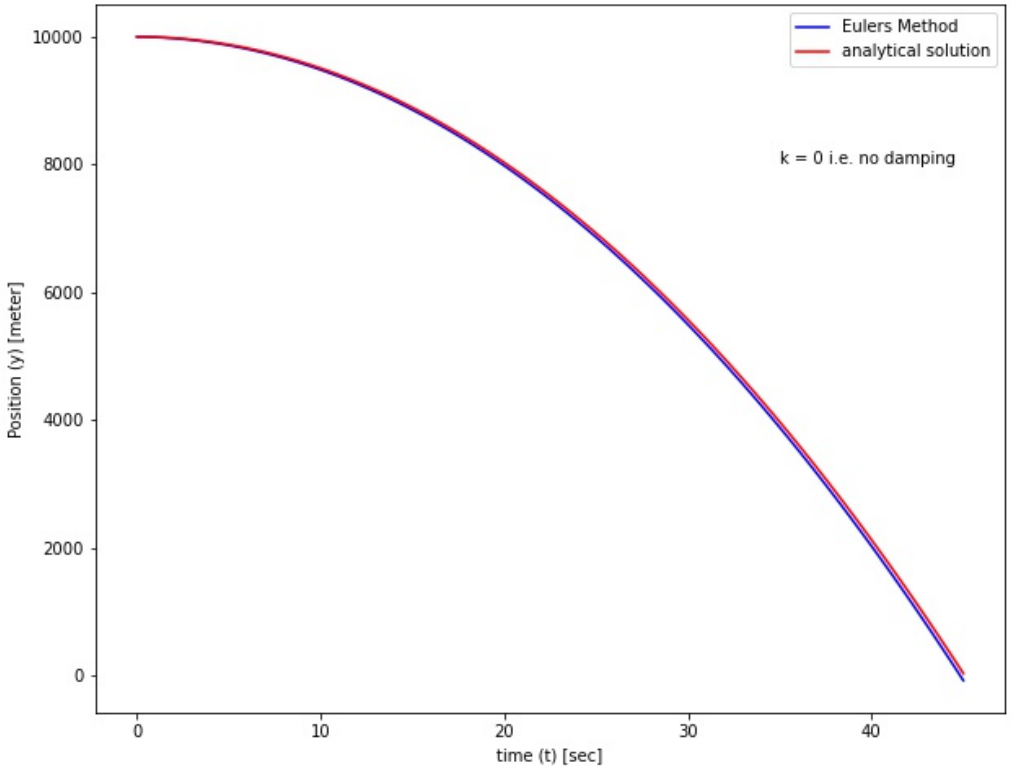

Figure 2: Comparison of Euler's Algorithm with Analytical Solution

2.2 Terminal Velocity

In case of velocity based damping, the object attains a final stable velocity, called terminal velocity, when the drag forces balances the weight of the body. The terminal velocity is given by:

$$v_{terminal} = \sqrt{\frac{mg}{k}} \tag{1}$$

For the given set of data, using equation(1), the terminal velocity can be calculated as:

$$v_{terminal} = \sqrt{\frac{(1\,kg)\cdot(9.81\,ms^{-2})}{(3\cdot 10^{-4}\,kg/m)}} = 180.82\,m/s$$

Using the Euler algorithm, the variation of velocity with time can be described using the following plot.

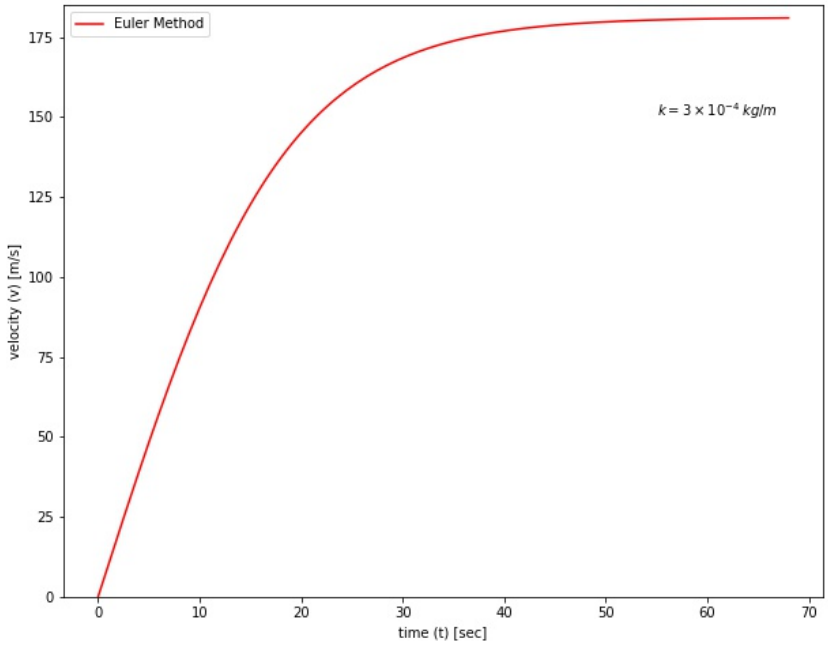

Figure 3: Variation of velocity with time [Euler method]

As seen in figure(3), the curve has asymptotic behaviour to line approximately y= 180 m/s. Indeed, from the data array of the calculated velocities, we find the following values of velocities for last 25 iterations:

Ind	Type	Size	
112	float	1	180.4460417610735
113	float	1	180.48508338511874
114	float	1	180.5220113532636
115	float	1	180.55693967536038
116	float	1	180.58997625592846
117	float	1	180.62122321640487
118	float	1	180.6507772008841
119	float	1	180.67872966613962
120	float	1	180.7051671566876
121	float	1	180.73017156562153
122	float	1	180.7538203819159
123	float	1	180.7761869248666
124	float	1	180.797340566307
125	float	1	180.8173469412094
126	float	1	180.8362681472549
127	float	1	180.8541629339281
128	float	1	180.87108688166603
129	float	1	180.8870925715682
130	float	1	180.90222974614855
131	float	1	180.916545461589
132	float	1	180.9300842319309
133	float	1	180.9428881656203
134	float	1	180.95499709480302
135	float	1	180.9664486977448
136	float	1	180.97727861473481

Figure 4: Values of Velocities in final iterations [Euler Method]

The value of terminal velocity as seen in the graph (3) and confirmed in figure(4) matches closely to the expected value as calculated using equation(1).

3 Euler-Richardson Algorithm

```
#Euler-Richardson Scheme
y, v, t = y0, v0, t0
y1arr, v1arr, t1arr = [y], [v], [t]
while y > 0:

    k1v = ac(y, v)*dt
    k1y = v * dt
    k2v = ac((y - k1y/2), (v-k1v/2))*dt
    k2y = (v - k1v/2)*dt
    v = v - k2v
    y = y - k2y
    t = t+dt

    t1arr.append(t)
    v1arr.append(v)
    y1arr.append(y)
```

Figure 5: Code Snippet for Euler-Richardson Algorithm

For the validation of the Euler-Richardson scheme, the same set of checks were performed as explained in section (2). The following plots were obtained:

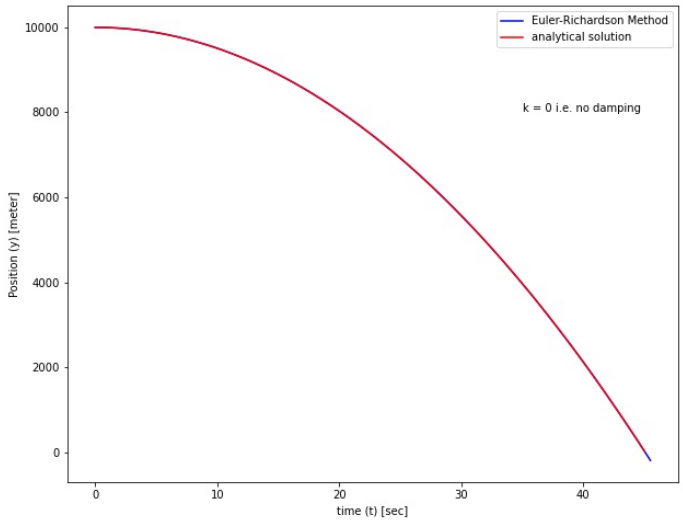

Figure 6: Comparison of Euler-Richardson Algorithm with Analytical Solution

As explained in section 2.1, the deviation from analytical solution is seen due to different values of acceleration due to gravity used in the scheme against the constant value used in analytical solution.

Similarly, figure (7) shows the plot obtained for the variation of velocity with time.

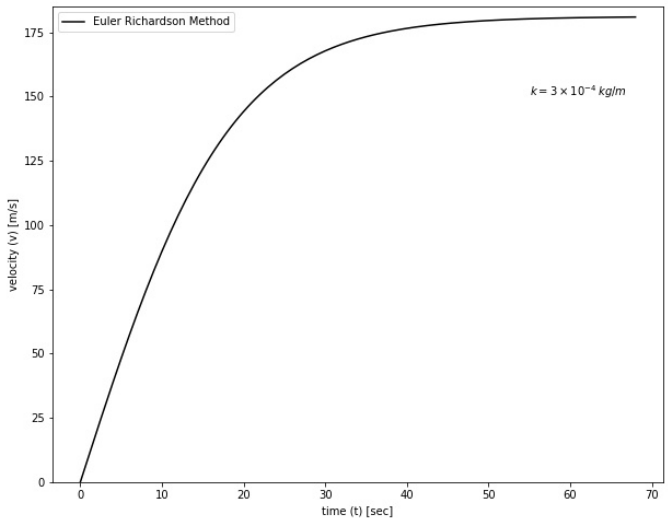

Figure 7: Variation of Velocity with Time[Euler-Richardson Method]

The obtained value of terminal velocity was in accordance with the expected value calculated using equation(1).

Ind	Type	Size	
112	float	1	180.34288156425615
113	float	1	180.38629880353798
114	float	1	180.4274300907598
115	float	1	180.46639530383837
116	float	1	180.5033080833581
117	float	1	180.5382761520293
118	float	1	180.57140161830952
119	float	1	180.60278126491735
120	float	1	180.6325068229423
121	float	1	180.66066523222517
122	float	1	180.68733888865748
123	float	1	180.7126058790215
124	float	1	180.73654020396714
125	float	1	180.75921198969615
126	float	1	180.78068768890054
127	float	1	180.80103027147726
128	float	1	180.82029940551925
129	float	1	180.83855162906002
130	float	1	180.85584051302754
131	float	1	180.87221681584293
132	float	1	180.88772863007858
133	float	1	180.9024215215722
134	float	1	180.91633866137357
135	float	1	180.92952095088438
136	float	1	180.94200714053295

Figure 8: Values of Velocities in final iterations [Euler-Richardson Method]

4 Comparison of Euler-Richardson and Euler's Scheme

The two schemes can be compared in respect to the time step dt used for the integration.

For smaller values of dt, the two schemes behave almost identical. However, for larger values, Euler-Richardson scheme is more stable than Euler scheme. This can be visualised by following the following plot of variation of velocity with time calculate at time steps $dt = 10.0\ sec.$

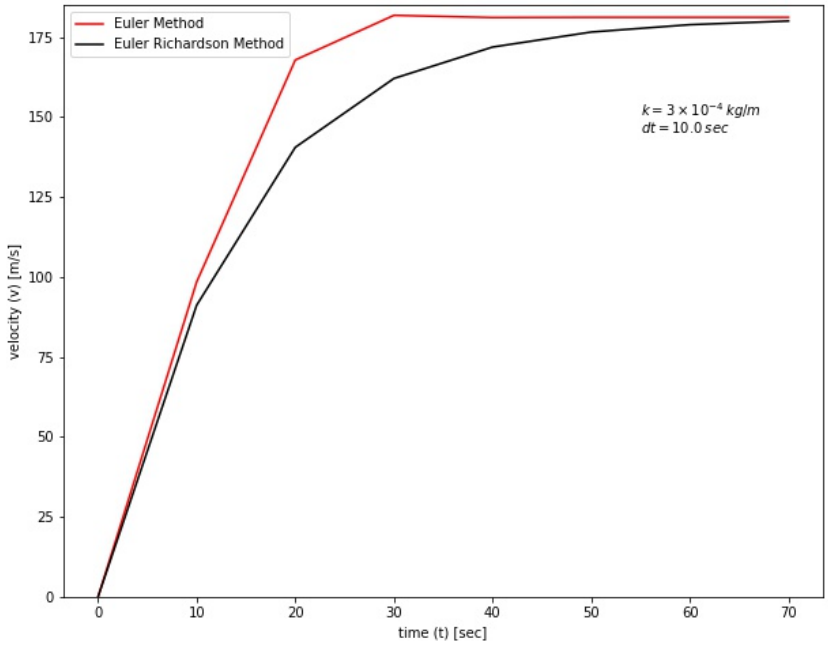

Figure 9: Velocity time graph for two schemes at dt= 10.0 sec

Here, the Euler-Richardson algorithm tends to get stable results only after about 7 iterations. At time steps, $dt = 6.5\, sec$, the Euler scheme tends to give stable results as well as seen in figure (10).

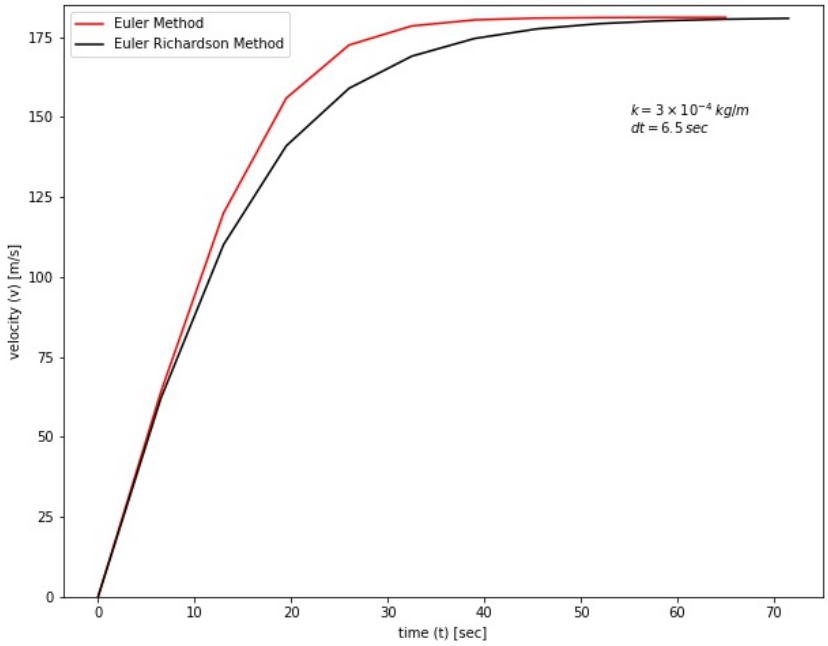

Figure 10: Velocity time graph for two schemes at dt= 6.5 sec

A
Variables Used

G = Universal Gravitational Constant = $6.67 \times 10^{-11}\, m^3 kg^{-1} s^{-1}$
M = Mass of Earth = $5.99 \times 10^{24}\, kg$
R = Radius of Earth = $6370\, km$
g = acceleration due to gravity at surface of Earth = $-9.81\, ms^{-2}$
m = mass of object = $1\, kg$
k = Coefficient of Drag force = $3 \times 10^{-4}\, kg/m$
y0 = initial position/height = $10000\, m$
dt = time step used in integration = $0.5\, s$ (except in Section 4)

B
Full Code

The general outline of the code looked like the following although some variables and plots were altered throughout the program to get desired graphs.

```
"""
Computational Physics
Project 1
@Lalit Chaudhary
l.chaudhary@jacobs-university.de
"""

import numpy as np
import matplotlib.pyplot as plt

#Setting up the constants
G = 6.67*10**(-11)
M = 5.99*10**(24)
R = 6370000
m = 1
k = 3*10**(-4)

#Initial conditions
t0 = 0
y0 = 10000
v0 = 0
g = -9.81

#height dependenet acceleration and velocity dependent damping
def ac(y, v):
   return  -1*(G*M*m)/(R*R*(1+y**2/R**2)) + (k/m)*v**2


dt = 0.5

#Euler method
y, v, t = y0, v0, t0
yarr, varr, tarr = [y], [v], [t]
while y > 0:

    a = ac(y, v)
    v = v - a*dt
    y = y - v*dt
    t = t+dt
    tarr.append(t)
```

```
    varr.append(v)
    yarr.append(y)


#analytical solution without damping
th = np.linspace(0, t, 100)
pos = []
for t in th:
    p = y0 + v0*t + 0.5 * ac(y0, v0) * t**2
    pos.append(p)


plt.figure(1, figsize=(10,8))
plt.plot(tarr, yarr, 'b-', label = 'Eulers Method')
plt.plot(th, pos, 'r', label = 'analytical solution')
#plt.text(35, 8000, 'k = 0 i.e. no damping')
plt.xlabel('time (t) [sec]')
plt.ylabel('Position (y) [meter]')
plt.legend()
#plt.savefig('NoDamping.jpg')


#Euler-Richardson Scheme
y, v, t = y0, v0, t0
y1arr, v1arr, t1arr = [y], [v], [t]
while y > 0:

    k1v = ac(y, v)*dt
    k1y = v * dt
    k2v = ac((y - k1y/2), (v-k1v/2))*dt
    k2y = (v - k1v/2)*dt
    v = v - k2v
    y = y - k2y
    t = t+dt

    t1arr.append(t)
    v1arr.append(v)
    y1arr.append(y)




plt.figure(1, figsize=(10,8))
plt.plot(t1arr, y1arr, 'b-', label = 'Euler-Richardson Method')
plt.plot(th, pos, 'r', label = 'analytical solution')
```

```
#plt.text(35, 8000, 'k = 0 i.e. no damping')
plt.xlabel('time (t) [sec]')
plt.ylabel('Position (y) [meter]')
plt.legend()
#plt.savefig('NoDampingER.jpg')


plt.figure(3, figsize=(10,8))
plt.plot(t1arr, y1arr, 'k-', label = 'Euler Richardson Method')
plt.plot(tarr, yarr, 'r-', label='Euler Method')
#plt.plot(th, pos, 'g-', label = 'analytical solution')
plt.xlabel('time (t) [sec]')
plt.ylabel('Position (y) [meter]')
plt.legend()
plt.text(55, 9200, 'dt = 0.5')
plt.show()


plt.figure(2, figsize = (10,8))
plt.plot(tarr, varr, 'r', label ='Euler Method')
plt.plot(t1arr, v1arr, 'k', label = 'Euler Richardson Method')
plt.xlabel('time (t) [sec]')
plt.ylabel('velocity (v) [m/s]')
plt.ylim(0, 185)
plt.text(55, 150, r'$k = 3 \times10^{-4} \:kg/m$')
plt.text(55,145, r'$ dt = 6.5\:sec$')
plt.legend()
#plt.savefig('stableE.jpg')
```