Fourth homework

Abstract: An application of Euler method.

Background:

1.3 It is often the case that the frictional force on an object will increase as the object moves faster. A fortunate example of this is a parachutist,the role of the parachute is to produce a frictional force due to air drag,which is larger than wold normally be the case without the parachute. The physics of drag will be discussed in more detail in the next chapter. Here we consider a very simple example in which the frictional force depends on the velocity. Assume that the velocity of an object obeys an equation of the form

$$\frac{dv}{dt} = a - bv$$

Where a and b are constants. You could think of a as coming from an applied force,such as gravity,while b arises from friction. Note that the frictional force is negative (we assume that b >0), so that it oppose the motion, and that it increase in magnitude as the velocity increases.Use the Euler method to solve $\frac{dv}{dt} = a - bv$ for v as a function of time. A convenient choice of parameters is $a = 10$ an $b = 1$ You should find that v approaches a constant value at long times,this is called the terminal velocity.

Text:

Euler method: $v(t+dt)=v(t)+(a-bv)dt$

Program in python:

```
import numpy as np
import matplotlib.pyplot as plt      #to import matplotlib's package

v=[ ]           #velocity
t=[ ]           #time
a=10            #assign a value to a
b=1             #assign a value to b
dt=0.1          #time step
v.append(0)     #assign a value to first item of v[ ]
t.append(0)     #assign a value to first item of t[ ]
end_time=9      #total time

for i in range(int(end_time/dt)):
    tmp=v[i]+(a-b*v[i])*dt   #Euler method
    v.append(tmp)   #add new value of v to v[ ]
    t.append(dt*(i+1)) #add new value of t to t[ ]
    print t[-1],v[-1] #print the value of v and t on each stap

plt.figure(figsize=(8,6))   #set the size of corresponding figure
plt.plot(t,v,label="v(t)",color="orange",linewidth=2) #plot the figure and set label and color and linewidth of the figure
plt.xlabel("t(s)")     #set the label of x axis
plt.ylabel("v(m/s)")   #set the label of y axis
plt.title("motion with fractional force") #set title
plt.ylim(0,12) #set the range of y axis
plt.legend()   #make it to show everything
plt.show()   #activate
```

Result:

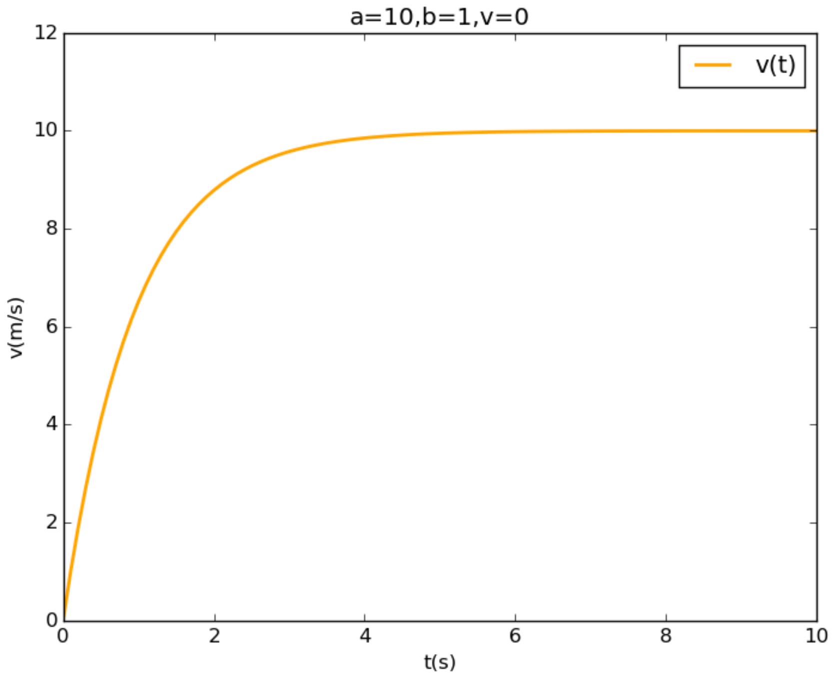
Data:

0.1 1.0
0.2 1.9
0.3 2.71
0.4 3.439
0.5 4.0951
0.6 4.68559
0.7 5.217031
0.8 5.6953279
0.9 6.12579511

1.0 6.513215599
1.1 6.8618940391
1.2 7.17570463519
1.3 7.45813417167
1.4 7.7123207545
1.5 7.94108867905
1.6 8.14697981115
1.7 8.33228183003
1.8 8.49905364703
1.9 8.64914828233
2.0 8.78423345409
2.1 8.90581010868
2.2 9.01522909782
2.3 9.11370618803
2.4 9.20233556923
2.5 9.28210201231
2.6 9.35389181108
2.7 9.41850262997
2.8 9.47665236697
2.9 9.52898713028
3.0 9.57608841725
3.1 9.61847957552
3.2 9.65663161797
3.3 9.69096845617
3.4 9.72187161056
3.5 9.7496844495
3.6 9.77471600455
3.7 9.7972444041
3.8 9.81751996369
3.9 9.83576796732
4.0 9.85219117059
4.1 9.86697205353
4.2 9.88027484817
4.3 9.89224736336
4.4 9.90302262702
4.5 9.91272036432
4.6 9.92144832789
4.7 9.9293034951
4.8 9.93637314559
4.9 9.94273583103
5.0 9.94846224793
5.1 9.95361602313
5.2 9.95825442082
5.3 9.96242897874
5.4 9.96618608086

5.5 9.96956747278
5.6 9.9726107255
5.7 9.97534965295
5.8 9.97781468766
5.9 9.98003321889
6.0 9.982029897
6.1 9.9838269073
6.2 9.98544421657
6.3 9.98689979491
6.4 9.98820981542
6.5 9.98938883388
6.6 9.99044995049
6.7 9.99140495544
6.8 9.9922644599
6.9 9.99303801391
7.0 9.99373421252
7.1 9.99436079127
7.2 9.99492471214
7.3 9.99543224093
7.4 9.99588901683
7.5 9.99630011515
7.6 9.99667010363
7.7 9.99700309327
7.8 9.99730278394
7.9 9.99757250555
8.0 9.99781525499
8.1 9.9980337295
8.2 9.99823035655
8.3 9.99840732089
8.4 9.9985665888
8.5 9.99870992992
8.6 9.99883893693
8.7 9.99895504324
8.8 9.99905953891
8.9 9.99915358502
9.0 9.99923822652

Corresponding figure:



If we change the initial value of v, say $v = 16$,we get

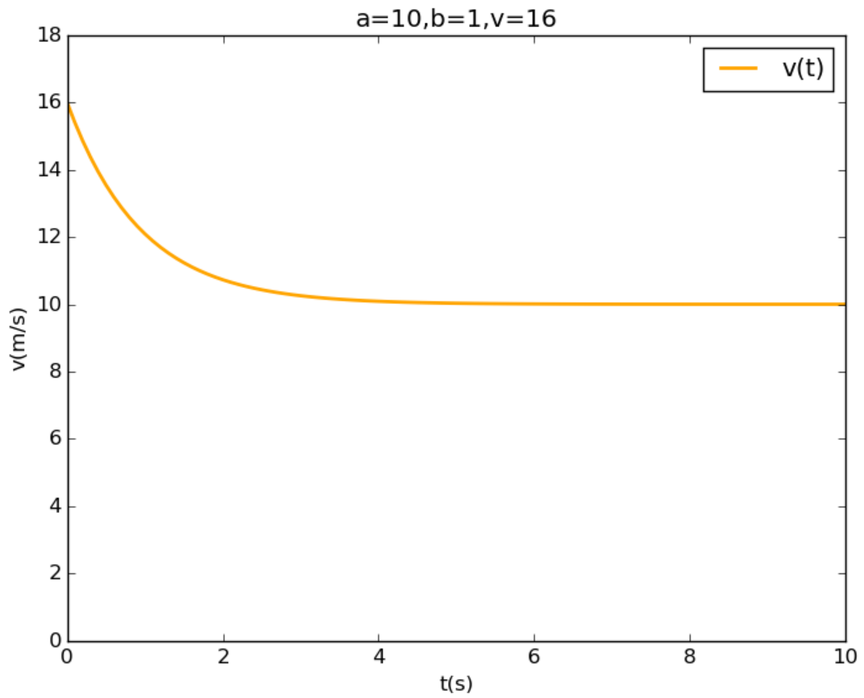
Data

0.1 15.4
0.2 14.86
0.3 14.374
0.4 13.9366
0.5 13.54294
0.6 13.188646
0.7 12.8697814
0.8 12.58280326
0.9 12.324522934
1.0 12.0920706406
1.1 11.8828635765
1.2 11.6945772189
1.3 11.525119497
1.4 11.3726075473
1.5 11.2353467926
1.6 11.1118121133
1.7 11.000630902
1.8 10.9005678118
1.9 10.8105110306
2.0 10.7294599275
2.1 10.6565139348

2.2 10.5908625413
2.3 10.5317762872
2.4 10.4785986585
2.5 10.4307387926
2.6 10.3876649134
2.7 10.348898422
2.8 10.3140085798
2.9 10.2826077218
3.0 10.2543469497
3.1 10.2289122547
3.2 10.2060210292
3.3 10.1854189263
3.4 10.1668770337
3.5 10.1501893303
3.6 10.1351703973
3.7 10.1216533575
3.8 10.1094880218
3.9 10.0985392196
4.0 10.0886852976
4.1 10.0798167679
4.2 10.0718350911
4.3 10.064651582
4.4 10.0581864238
4.5 10.0523677814
4.6 10.0471310033
4.7 10.0424179029
4.8 10.0381761126
4.9 10.0343585014
5.0 10.0309226512
5.1 10.0278303861
5.2 10.0250473475
5.3 10.0225426128
5.4 10.0202883515
5.5 10.0182595163
5.6 10.0164335647
5.7 10.0147902082
5.8 10.0133111874
5.9 10.0119800687
6.0 10.0107820618
6.1 10.0097038556
6.2 10.0087334701
6.3 10.0078601231
6.4 10.0070741107
6.5 10.0063666997
6.6 10.0057300297

6.7 10.0051570267
6.8 10.0046413241
6.9 10.0041771917
7.0 10.0037594725
7.1 10.0033835252
7.2 10.0030451727
7.3 10.0027406554
7.4 10.0024665899
7.5 10.0022199309
7.6 10.0019979378
7.7 10.001798144
7.8 10.0016183296
7.9 10.0014564967
8.0 10.001310847
8.1 10.0011797623
8.2 10.0010617861
8.3 10.0009556075
8.4 10.0008600467
8.5 10.000774042
8.6 10.0006966378
8.7 10.0006269741
8.8 10.0005642767
8.9 10.000507849
9.0 10.0004570641
9.1 10.0004113577
9.2 10.0003702219
9.3 10.0003331997
9.4 10.0002998797
9.5 10.0002698918
9.6 10.0002429026
9.7 10.0002186123
9.8 10.0001967511
9.9 10.000177076
10.0 10.0001593684

Corresponding figure



Conclusion:

No matter v is lager or smaller than 10 , it will finally approch 10.This is consistent with analysis.

So Euler method works well. And numerical computation is a powerful methodology.