Numerical approach to analyse physical system

Industrial training (7PH192)

By

SHASHI KUMAR SAMDARSHI

Roll No: 1510002
Under the guidance of
Dr. Anurag Sahay
Department of Physics
National Institute of Technology Patna

Outlines

1. Introduction to Euler's method, Euler-Cromer method
2. Physical system (Simple pendulum, damped and a damped, driven)
3. Summary
4. References

Euler’s method and Euler-Cromer method

Euler’s method is a numerical method to solve first order differential equation with given initial value. There is a problem with Euler’s method for oscillatory motion who solved it is Cromer.

Physical system

1. Simple pendulum ,with damping and with damping and forcing

Simple pendulum

The swinging of a pendulum is governed by the equation

$d^2 x/dt^2 + (g/L) \sin(x)=0$

x=theta, y=omega

Euler scheme

(i) Plot of x versus time (using Euler scheme)

Euler-Cromer scheme

(ii) Plot of x versus time (iii) Plot of x versus y

With dmping

$d^2 x/dt^2 + (g/L) \sin(x)+ß*dx/dt=0$

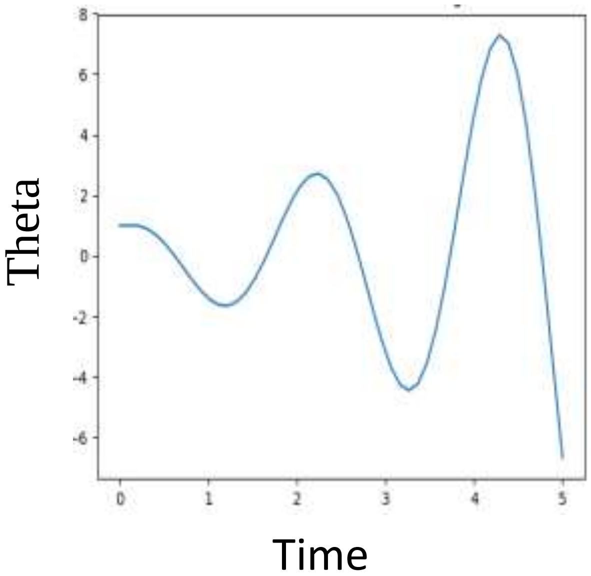
ß=damping parameter

(iv) Plot of theta versus time (v) Plot of Omega versus time

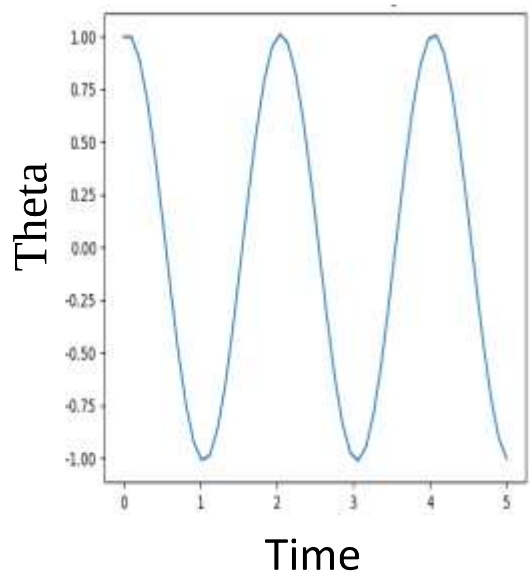
With damping and driving force

$d^2 x/dt^2 + (g/L) \sin(x)+ß*dx/dt=A\cos(wt)$

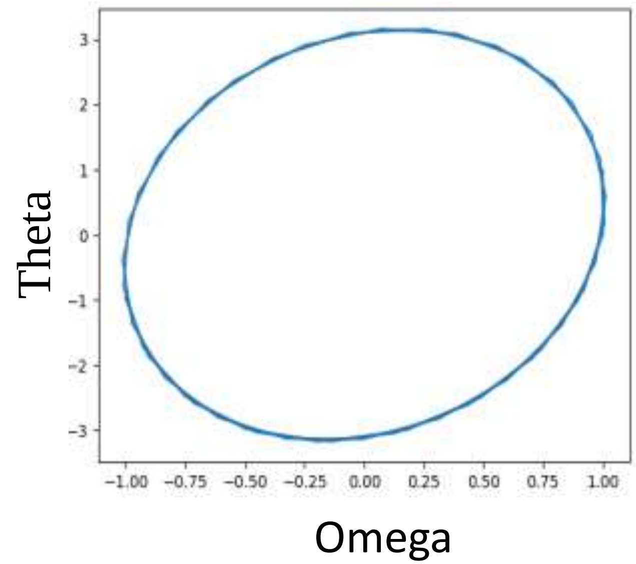
(vi) Plot of theta versus time



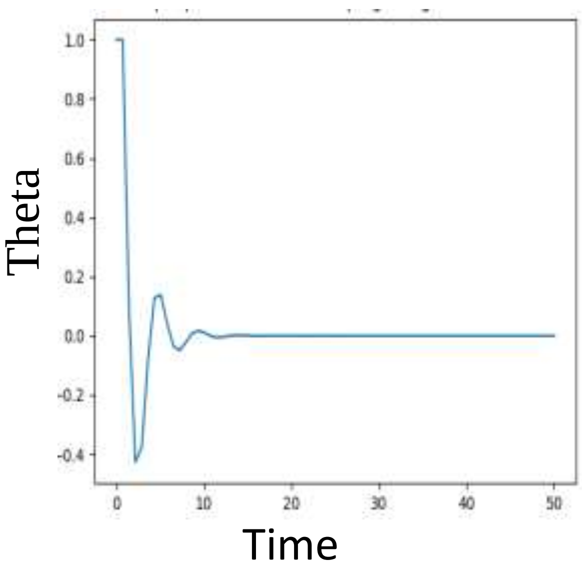
(i)



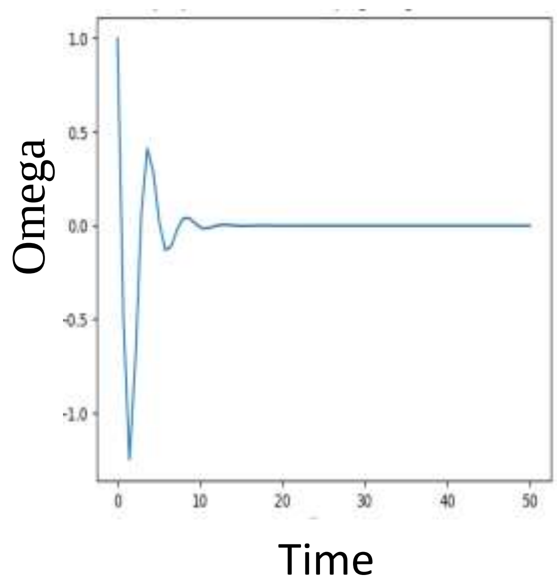
(ii)



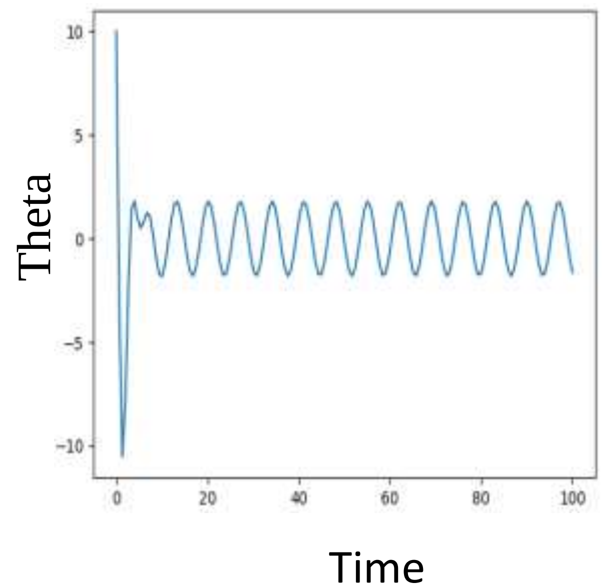
(iii)



(iV)



(v)



(vi)

Summary

- For any oscillatory motion , the energy of Euler's solution increases with time.
- Problem with Euler's solution is fixed by Cromer by the position x_i and velocity v_i at step i to find the position x_{i+1}.

References

1. Rubin H. Landau Manuel J. Paez Cristian C. Bordeianu Computational Physics
2. E. Ward Cheney, David R. Kincaid Numerical Mathematics and Computing, Sixth Edition
3. S S SASTRY Introductory methods of numerical analysis
4. Paul L DeVries A first course in computational physics
5. Steven E. Koonin Computational physics,

Code

```
import numpy as np
from matplotlib import pyplot as plt
l=1,g=1,t0=0,x0=1,y0=1,tf=5,n=50
deltat=(tf-t0)/(n-1)
t=np.linspace(t0,tf,n)
y=np.zeros([n])
y[0]=y0
x=np.zeros([n])
x[0]=x0
for i in range(1,n):
    y[i]=y[i-1]+deltat*x[i]  #y=theta
    for i in range(1,n):
        x[i]=x[i-1]-(g/l)*y[i-1] #x=omega
for i in range(n):
    for i in range(n):
        print(y[i],x[i])
        plt.xlabel("time")
        plt.ylabel("omega")
        plt.plot(t,x)
        plt.show()
```

Thank you!