Oscillation Homework:

Algorithm

The algorithm used for this project were:

For Euler-Cromer Method:

For Euler Method:

Where:

– Angular Velocity

– Gravity

– Length

– Angle

– Change in Time

– Energy

– Mass

The difference in the equations is that the Euler-Cromer Method uses the term to find while the Euler Method uses the term to find .

Code

#include <iostream>

#include <cmath>

#include <fstream>

#include <iomanip>

using namespace std;

// Problem 3.1 - Energy and the Euler-Cromer Method.

double EulerCromerMethod()

{

// Declaring the values. Note that theta has to be converted to radians.

long double AngularVelocity = 0, Gravity = 9.81, Length = 1, dt = .01, T = 1, Pi = 3.1415926535898, Theta = 3 \* (Pi/180), Time = 0, Mass = .5, Energy = .006722, TimeInterval = 1000,

AngularVelocity\_i = 0, Theta\_i = 3 \*(Pi/180), Brackets = 0, dE = 0, GdivL = Gravity / Length;

// Opening the file that will write the data for the project.

ofstream EulerCromer;

EulerCromer.open("Euler-Cromer.txt");

EulerCromer << "Euler-Cromer Method\n\n";

// Fixing the precision.

EulerCromer << fixed << showpoint << setprecision(6);

// Creating the header and writing the first value. Note that the angle was converted back to degrees.

EulerCromer << "Time (s) Angle (deg) Velocity Energy" << endl;

EulerCromer << Time << " " << Theta \* (180/Pi) << " " << AngularVelocity << " " << Energy << endl;

// Loop to calculate Euler-Cromer Method

while (T <= TimeInterval)

{

// This method already takes care of the omega i+1 case.

AngularVelocity = AngularVelocity - (Gravity / Length) \* Theta \* dt;

Theta = Theta + AngularVelocity \* dt;

Time = Time + dt;

// Since the values currently held in the program are the i+1 values, they must be changed back to the i values for the Energy equation.

Theta\_i = Theta - AngularVelocity \* dt;

AngularVelocity\_i = AngularVelocity + (Gravity / Length) \* (Theta\_i)\* dt;

// Broke up the Algorythim for sake of clarity.

Brackets = (pow(AngularVelocity\_i, 2) - GdivL \* pow(Theta\_i, 2));

dE = .5 \* GdivL \* Brackets \* pow(dt, 2);

Energy = Energy + dE;

// Displaying the results.

EulerCromer << Time << " " << Theta \* (180/Pi) << " " << AngularVelocity << " " << Energy << endl;

T++;

}

// Close the file.

EulerCromer.close();

return 0;

}

double Euler()

{

// Same thing as above.

long double AngularVelocity = 0, Gravity = 9.81, Length = 1, dt = .01, T = 1, Time = 0, Mass = .5, Energy = .006722, TimeInterval = 1000,

Pi = 3.1415926535898, Theta = 3 \* (Pi / 180), Theta\_i = 0, AngularVelocity\_i,

MGL = Mass \* Gravity \* Length, Brackets = 0, dE = 0;

// Opening the file.

ofstream Euler;

Euler.open("Euler.txt");

Euler << "Euler Method\n\n";

// Fix the output.

Euler << fixed << showpoint << setprecision(6);

// Create headers.

Euler << "Time (s) Angle (deg) Velocity Energy" << endl;

Euler << Time << " " << Theta \* (180/Pi) << " " << AngularVelocity << " " << Energy << endl;

// Loop calculating Euler Method.

while (T <= TimeInterval)

{

Theta = Theta + AngularVelocity \* dt;

// Since the angular velocity term uses theta i not theta i+1, the following program must be used.

Theta\_i = Theta - AngularVelocity \* dt;

AngularVelocity = AngularVelocity - (Gravity / Length) \* ( Theta\_i) \* dt;

Time = Time + dt;

// Same issue with the theta i vs theta i+1

AngularVelocity\_i = AngularVelocity + (Gravity / Length) \* (Theta\_i) \* dt;

Brackets = (pow(AngularVelocity\_i, 2) + (Gravity / Length) \* pow(Theta\_i, 2));

dE = .5 \* MGL \* Brackets \* pow(dt, 2);

Energy = Energy + dE;

// Output the values.

Euler << Time << " " << Theta \* (180/Pi) << " " << AngularVelocity << " " << Energy << endl;

T++;

}

// Close the file.

Euler.close();

return 0;

}

int main()

{

// Run the subprograms.

EulerCromerMethod();

Euler();

}

How to Run the Code

This code is written in C++ so in order to run it, the g++ compiler should be used. This compiler should already be in Omega so all needed to be done is to run it.

Results and Analysis

# Euler-Cromer Method

Using the Euler-Cromer Method, it can be seen that the angle oscillates between three and negative three, but it does so constantly and uniformly.

The energy of the system seems to oscillate as well but since this is an approximation and in real life there is supposed to be a conservation of energy, the oscillation averages to be centered at the initial energy value so it is close to conservation of energy.

# Euler Method

It can be seen that over time, the angle that the Euler method gives at each time interval oscillates bigger and bigger as time goes on. This is strange because it suggest that the pendulum goes higher and higher than the original height.

Because of the strange nature of the angle’s value, the energy of the system slowly gets bigger and bigger. This is in conflict with the concept of the conservation of energy.

Conclusion

When it comes to calculating the motion of a pendulum, the Euler-Cromer Method is superior to the Euler Method since the Euler-Cromer Method obeys the conservation of Energy principle better than the Euler Method and the results the Euler-Cromer Method gives make more physical sense.