Nuclear Decay Homework:

Algorithm

The starting point of this program was the following equation:

Where:

– Amount of Uranium.

– The derivative in respect to time.

– Time constant.

Through manipulation becomes:

Since at , the following:

To see if this is the result that we get, the Euler Method will be used to see if the following shows to be true.

Where:

– Amount of Uranium after a time step.

– Amount of Uranium before the time step.

– Time step.

Code

// Nuclear Decay Program.

// Here are all the libraries that will be used for the project.

#include <iostream>

#include <cmath>

#include <fstream>

using namespace std;

// Here is the fucntion where the Euler Method will be performed.

void Decay(double TimeConstant, double InitialAmount, double dt)

{

// Here the file which will contain all the data points will be created.

ofstream NuclearDecay;

NuclearDecay.open("Nuclear.txt");

// Here is iwhat is going to be outputted.

double amount, time = 0;

// Let the starting amount be set.

amount = InitialAmount;

// Header for the data points.

NuclearDecay << "Time Amount" << endl;

// Amount at time 0.

NuclearDecay << time << " " << amount << endl;

// Loop where the Euler method will be done.

// The total amount of time this will be done is 5 seconds, so dividing it by the time interval will give how many itarations need to be complete.

for (int t = 0; t < 5/ dt ; t++)

{

// Euler Method.

amount = amount - (amount / TimeConstant) \* dt;

// increase the time display counter.

time = time + dt;

// Display the time and amount at the time.

NuclearDecay << time << " " << amount << endl;

}

// Display the starting and ending amounts to the user.

cout << "The amount of Uranium that we started with was " << InitialAmount << " nuclei.\nAfter " << time << " seconds, " << amount << " nuclei was left." << endl;

// Close the program.

NuclearDecay.close();

}

int main()

{

// For a decay constant of 1, starting amount of 100, and .01 for the time step.

Decay(1, 100, .01);

// End program.

return 0;

}

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. The file extension that seemed to work best is the .C extension. Note that the file creates files but in terms of user interaction, the program only lets the user know what the program does and when it completed the run.

Results and Analysis

Once the program was completed, the following graph was created using the data obtained via the Euler equation. The program outputted the data points to a file and from the file, the data was pasted to Excel and was plotted using its plot capabilities.

The time step used was .01 because it needed to be decently small in order for the program to be close to what it needed to be. The time of the entire simulated process was 5 seconds because at that point the values were all really small. The starting amount of 100 was just chosen because it was what the book used.

The Graph below has the equation that excel fit to it as well as the value. The fact that the value is 1 means that the graph’s fitting was pretty accurate to all the data points.

The theory predicted that for a initial amount of 100 and a decay constant of 1:

The trend line gives us:

And the error in the time constants is:

Conclusion

The Euler method is a great tool to calculate certain differential equations numerically with great accuracy. The reason why the Euler method was not exact was because the time step is not infinitesimally small, which is called for in calculus, and that is why the Euler method is sensitive to the time interval that is used.