Sophie Ardell

October 28, 2015

Computational Physics

Homework Set 6

**Problem 1**

This code implements the implicit and explicit Euler method to integrate the differential equation:

from *x* = 0 to *x* = 1 in *n* equal steps (so *h* = ). The code uses = 100, and the initial condition *y* = 1 at *x* = 0 throughout all parts of the code.

The explicit equation for any *ym* is:

The implicit equation for any *ym* is:

The analytic equation for any *ym* is:

For Part A, I held *n* = 1000, and then graphed the results for the implicit, explicit, and analytic equations. See attached image.

The y-values drop quickly to near 0 (final values are approximately 10-42) so I’ve also attached the data to this report. The all the lines match up quite well, though near the end, the implicit and explicit differ from the analytic values by about a factor of , but the shapes follow the same form.

I’ve made a graph of the relative error at each point as well in order to better compare functions, which I have also attached below. It is interesting to note that relative error stays constant for explicit error while implicit error becomes increasingly worse as x increases.

For Part B, I found the minimum number of steps for the explicit Euler method by using this equation:

from my notes, which holds true for any positive Because is held constant for this problem, the answer is simply that h must be less than 0.02 in order for the function to be stable. Because *h* = , where n is the amount of steps, n must be greater than 50.

For part C, I ran my code for n = 49, 50, and 51. I’ve attached the graphs to this write-up. The data makes sense with my understanding of how the function is supposed to behave. At *n* = 49, the y-values are becoming increasingly larger and do not follow the analytical function at all, therefore it is not stable. This makes sense since it is less than 50. At 50, the y-values oscillate between 1 and -1, indicating a turning point. At 51, the y-values are becoming increasing smaller over time, indicating that they will eventually converge, therefore it is stable.

Macintosh HD:Users:Sophie:Desktop:Physics 302:Homework 6:N1000.ps

Macintosh HD:Users:Sophie:Desktop:Physics 302:Homework 6:Relative Error for N1000.psMacintosh HD:Users:Sophie:Desktop:Physics 302:Homework 6:N51.ps

Macintosh HD:Users:Sophie:Desktop:Physics 302:Homework 6:N50.psMacintosh HD:Users:Sophie:Desktop:Physics 302:Homework 6:N49.ps

**Problem 2**

For parts B, C, and D, this program simulates the equation of motion for a single metronome, which is as follows:

[1]

where is the angle that the pendulum makes with the vertical. The second term describes the (non-linear) restoring force, and depends on the pendulum’s mass *m*, the distance rCM of its center of mass from the pivot point, the pendulum’s moment of inertia *I,* as well as the gravitational acceleration *g*. The final term model’s the metronome’s escapement; for , where is some constant, the term acts as a driving force, where as for , it acts like a damping term. For small values of the constant , this term results in oscillations of an amplitude of approximately .

*Part B:*

When and small , equation[1] results in simple harmonic motion.

Allow . The form of a simple harmonic oscillator is as follows:

Furthermore, when is small, . Equation[1] can be rewritten as:

This is in the same form as a simple harmonic oscillator.

If the pendulum is of mass *m* on a massless string of length *l*, and given that (where ), and given that the period should be as follows:

which is the period exactly from Problem 5.1.

*Parts C and D:*

To create a program that solved this equation, I broke up the equation into simpler components. Let in equation[1]. This leads to two easier to solve first-order ODE’s:

which you can solve independently, then bring together to integrate and find the solution for the next value. This occurs over an interval of time, which I set to be from 0 to 1000, in N steps, which I set as 1000 for this project. I have attached a graph of how this behaved over 100 intervals since a larger range was very difficult to read on a graph. I used and I used both and . Given that is small, the amplitude should be 2. From the graph, the amplitude appears to be approximately 0.2, or 2, which makes sense!

I included a point where each period ended on the graph as well. I had expected when to have half the period of when , as I put their periods in a ratio. However the data does not reflect this, as there seems to be about 10 periods of to one period of when .

*Part E:*

I am now considering a system of two metronomes, described by and . For each equation of motion, I now include a coupling term, which is for as follows:

and similar for The code is pretty similar to the *Part C* and *D*, except that I have created two instances of pendulums that are now dependent on the other’s current value. At each interval, each pendulum object is evaluated like in the previous method, but then I make sure to update the value of the other metronome’s value after the ODE’s are evaluated. For both metronomes, I used . The two oscillators are initially out of phase, with one of them at rest with 0 displacement, and the other with an initial “push”. I used I’ve iterated over 1500 intervals, but the data looks essentially the same from 500 and beyond (in that that both functions of are perfectly aligned), so I chose to show from 450 to 550 for the later intervals in order to reflect the function accurately as it trends towards synchronization.

Macintosh HD:Users:Sophie:Desktop:Physics 302:Homework 6:Problem 2:partEearly.psMacintosh HD:Users:Sophie:Desktop:Physics 302:Homework 6:Problem 2:partElate.psMacintosh HD:Users:Sophie:Desktop:Physics 302:Homework 6:Problem 2:partEmid.psMacintosh HD:Users:Sophie:Desktop:Physics 302:Homework 6:Problem 2:partC.ps