ME140A - Homework 3

Due by 11:59PM, Oct 28th, by email to ameiburg@ucsb.edu. Collaboration is encouraged!

1 (Almost) Exponential Decay

Consider a sample of a nuclear isotope decaying over time, $y(x)$. It has a basic rate of decay proportional to its own amount y. But when there's a large quantity, the neutrons it gives off hit more of the sample, accelerating the decay by a factor $1 + y$. We can model this as follows:

$$y'(x) = -(1 + y)y$$

If we have a quantity of 5 units of the substance, this becomes an initial condition

$$y(0) = 5$$

(a) Solve this equation exactly, including the intial condition. It is a separable equation. You're free to use computer algebra systems such as Wolfram to help you solve this.

(b) Write MATLAB code to solve this numerically over the interval $x = 0$ to $x = 4$. Use the Euler Method,

$$y_{n+1} = y_n + f(x_n, y_n)h$$

with $h = 0.03$.

(c) Use your data from part **(b)** to estimate when the quantity of the isotope, y, drops to a safe level of 0.04. Then compute the exact time with your equation from **(a)**. How accurate is your estimate?

(d) Run your same code form part **(b)** but with a larger step size of $h = 0.3$. What happens? Explain.

(e) Modify your code from **(b)** to instead use the predictor-corrector method,

$$z = y_n + f(x_n, y_n)h$$

$$y_{n+1} = y_n + \frac{f(x_n, y_n) + f(x_n, z)}{2}h$$

Optimize your code by making sure that you **only use two function evaluations per step**. Use $h = 0.03$.

(f) Again compute the point where $y(x)$ drops below 0.04, with your predictor-corrected method. Compare with part **(c)**. How does the accuracy compare with the Euler Method?

2 Adaptive Step Sizes

This is a continuation of Problem 1. You'll need to solve at least up to **1(b)** first. It is separated out here to organize the ideas.

We saw that having h too large makes the simulation unstable. We know that $h = 0.03$ works okay for this setup, but what if $y(0) = 5000$ instead? We would need to take a very small step size, like $h = 0.00003$. But then, at later times in that simulation, once $y(x)$ is very small, these small steps would make our solver take a very long time and waste a lot of memory.

We solve this by adding *adaptive step sizing* to our Predictor-Corrector method. Start with an aggressive step size, such as $h = 0.1$. We get two different estimates for the derivative, $f(x_n, y_n)$ and $f(x_n, z)$. If these values are too different, then we cut h in half and try again (and again and again). Specifically, we'll compute

$$d_1 = f(x_n, y_n)$$

$$d_2 = f(x_n, z)$$

and then we proceed only if

$$|d_2 - d_1| < 0.1 \max(|d_2|, |d_1|)$$

or

$$|d_2 - d_1| < 0.0001.$$

The first is a relative error of 0.1, the second is an absolute error. Once the error is small enough, we take a step. On the next step, we set h back to 0.1 (and then may have to reduce it again).

Your job is to implement this, and use it to solve the equation above with $y(0) = 1000$. Make sure to save the x coordinates as well as the y coordinates, because they will be irregularly spaced. Display (somehow – with some kind of plot, it's up to you) how the step size varies over the course of the simulation.