DIFFERENTIAL EQUATIONS

Most students will take an entire course on solving differential equations. Most if not all students take their differential equations course after computer programming, so discussing programming in the context of the math course has little value. However, comparisons will be made between the two to demonstrate the utility of the numerical method.

One of the simplest differential equations is a first order system (FOS) meaning that the first derivative is the highest derivative in the equation:

$$\frac{dy}{dt} = my$$

In this expression, *m* is a constant and *y* is a function of *t*. In differential equations courses, students spend a considerable amount of time learning how to identify an equation for *y* that is a function of *t* only that satisfies the differential equation. For the above FOS, a function whose derivative is equal to a constant times the function is sought. The solution ends up being the following:

$$y = y_{init}e^{mt}$$

To get this solution, the equation is integrated from $t = 0$ to *t* and y_{init} is the value of *y* at $t = 0$. By the time the student reaches senior year, this calculation is trivial. Keep in mind that tougher differential equations exist and numerical methods will be better suited to solve them.

In this tutorial, a numerical method will be used to calculate the relationship between *y* and *t*. It is important to keep in mind that a discrete data set relating *y* and *t* will be found rather than explicit function. It is also important to keep in mind that the solution will be approximate.

There are many numerical methods that can be used to solve this equation. The easiest is called the Euler Method. It is based on the fact that the derivative can be rewritten as the following:

$$\frac{dy}{dt} = \frac{y_{i+1} - y_i}{\Delta t}$$

This is called the discretization of the derivative. Instead of taking limits, finite values are used. The value, y_{i+1}, is the value of the function *y* that is Δt in time away from y_i. Each step in Δt is used to calculate the next value of *y*. Our equation for the FOS using the Euler Method is:

$$\frac{y_{i+1} - y_i}{\Delta t} = my_i$$

Now to implement the Euler Method, y_{i+1} is solved for:

$$y_{i+1} = y_i + \Delta t m y_i = y_i(1 + m\Delta t)$$

If we know y_1, which is an initial condition, the next value, y_2, can be calculated based on a small change in t. This value of y_2 will be the value of the function at $t = \Delta t$. The next value y_3 can be calculated based on y_2. y_3 is the value of the function at $t = 2\Delta t$. The method continues to step in time calculating a corresponding y at each t. In the end, the t versus y relationship is a set of discrete data rather than a function for y.

By now, the student should be capable of writing pseudo-code to solve the FOS. Some programming components that will be needed include indexing and looping. The Euler Method code often looks like a melting of numerical integration code and root finding code.

The true power of the numerical method comes from the ability to handle two other scenarios quite easily. The first scenario is where the right hand side of the first equation in this tutorial is a function of both y and t. The example considered here is:

$$\frac{dy}{dt} = tye^{-5t}$$

The author is not even sure that an exact solution for this equation exists. However, using the Euler Method the above equation can be converted into the following equation:

$$\frac{y_{i+1} - y_i}{\Delta t} = t_i y_i e^{-5t_i}$$

Then this can be rearranged to solve for y_{i+1}:

$$y_{i+1} = y_i + t_i y_i e^{-5t_i}\Delta t$$

The next value of y depends on the previous value of t and y in this case. A slight change from the FOS, but something that is no more difficult to implement. The student will notice that if the differential equation looks like:

$$\frac{dy}{dt} = f(t, y)$$

Then the Euler Method solution for the next value of y, y_{i+1}, is:

$$y_{i+1} = y_i + f(t_i, y_i)\Delta t$$

The difficulty of finding the Euler Method solution does not change depending on the function on the right hand side of the equation.

The second scenario that is made easier by numerical methods is higher order derivatives, which will be similar to having multiple differential equations to solve simultaneously.

Consider the second order differential equation below:

$$\frac{d^2y}{dt^2} = a\frac{dy}{dt} + by + c$$

The discretization of the second derivative is not as straight forward as the first derivative. In the Euler Method, it will be avoided altogether by defining a second variable. For this variable, *z* will be used and set equal to the first derivative of y. By doing so, the single second order differential equation can be converted into two first order systems:

$$\frac{dy}{dt} = z$$

$$\frac{d^2y}{dt^2} = \frac{dz}{dt} = az + by + c$$

Then the two equations must be discretized and rearranged to get the following:

$$y_{i+1} = y_i + z_i\Delta t$$

$$z_{i+1} = z_i + (az_i + by_i + c)\Delta t$$

Now there are two equations to be solved. Both depend on the previous values of each other. As long as two initial conditions are give, one for *y* and one for *z*, the problem can be solved quite easily. This can be carried out for even higher order derivatives. A 6th order derivative would be converted into six first order differential equations, which would require 6 initial conditions.

In general, for all types of differential equations discussed in this tutorial, the Euler Method will work well using the same form of solution to solve all of these problems. Finding exact solutions is not nearly as straightforward, if they can be found at all. The student should keep in mind that the Euler Method is not perfect. It does suffer from relatively large errors compared to other techniques. And, oh yeah, there are many other techniques to solve ordinary differential equations. In a similar fashion to numerical integration, there are an infinite number of solution techniques for differential equations.

Finally, pseudo-code and MATLAB code has been completely omitted from this tutorial on purpose. The student, at this point in the semester, should be able to write their own pseudo-code and MATLAB code. If difficulties arise, keep in mind that the MATLAB code is very similar to numerical integration and root finding code.