Collin Hitchcock

Comp Phys Chapter 1.1 → 1.6 notes

- Many physics problems contain ODEs, computational techniques are used to help construct functions capable of solving them
- A simple introduction: radioactive decay. The process of radioactive decay is volatile meaning it is hard to predict precisely. Therefore answers are provided as probabilities / approximations. This is where Euler's method comes in.
- Differential equations can be solved without a numerical solution which is more beneficial from a computational standpoint (functions) Usually ends up as an initial value problem to get one variable in terms of another (N in terms of time [t] where t = 0)
- Most ODEs can be boiled down to using the Taylor expansion:

$$N_U(\Delta t) \;=\; N_U(0) \;+\; \frac{dN_U}{dt}\,\Delta t \;+\; \frac{1}{2}\,\frac{d^2 N_U}{dt^2}\,(\Delta t)^2 \;+\cdots\,, \qquad (1.3)$$

- Our initial value problem starts with $N_U(0)$ since that's at t = 0. If the change in t is relatively small then its better to leave off the higher power terms which leaves us with:

$$N_U(\Delta t) \;\approx\; N_U(0) \;+\; \frac{dN_U}{dt}\,\Delta t\,. \qquad (1.4)$$

- Assuming that delta t is small and nonzero, then you can say:

$$N_U(t+\Delta t) \;\approx\; N_U(t) \;+\; \frac{dN_U}{dt}\,\Delta t\,, \qquad (1.6)$$

- However, the results will only be approximations and should not be taken at face value. This means that the fewer data points you use, the less accurate the approximation becomes. So by adding as many data points possible will narrow down the approximation to be more accurate
- Programming is similar to handwriting in the sense that there are methods to make it more legible for others to understand it. Its easier for the programmer to know what their function does, but others may struggle to connect the not so obvious dots.

- There are three subroutines that typically take place, initialize, calculate and store. Each doing almost exactly what it sounds like. Initialize sets up the variables with their initial values. The calculate function then uses the Euler method and set conditions to make approximations based on the initial values and conditions. Then the store function could be placing the results into files or graphs to use later.
- The most crucial point is plotting the data and doing that properly. If you dont know what values you are plotting, its likely you dont know how to plot them either.
- Another important step is to check the outputs by simply printing them out as you go. Knowing what the result is supposed to be ahead of time can help with debugging.
- You can also check if you answer is reasonable by changing the step sizes. Granted you want to use a lot of steps for a more refined approximation, but if you were to increase/decrease the amount and it drastically changes the output of the graph/data, then there is likely something wrong with the approximation.
- Its also important to take error into consideration as changing the steps for radioactive decay shows the same pattern, but a change in approximation for the number of nuclei left. This shows that there is some degree of error with the approximation so by adding a “global error” you can account for some of it. This is proportional to the product of the number of time steps and error per step (Δt)^2
- Overall its good practice to add comments to the code you’re writing and when trying to interpret whether or not the data looks right, try plotting it. Still unsure, then try different time steps. If something seems off, then print out from the potential area causing the issue.