Reading Notes - 1.1-1.6

1.1 - Radioactive Decay

- Definition: When unstable nuclei lose energy by radiation. Must be done through probability and average times for decay.

$$N_U = N_U(0)\, e^{-t/\tau} ,$$

Nv(0) = number of nuclei present at t = 0
Tau = time constant

1.2 - A Numerical Approach

- Taylor expansion for Nv:

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

- Euler Method: Useful general algorithm for solving ordinary differential equations.
- Videos I found because Euler is still so confusing for me:
  - Calculus explanation: https://www.youtube.com/watch?v=ukNbG7muKho&t=30s
  - Python Help: https://www.youtube.com/watch?v=YKHNZk7vi-k

- Note that Euler is not the end-all-be-all and all algorithms have their own strengths and weaknesses.

1.3 - Design and Construction of a Working Program: Codes and Pseudocodes

- Pseudocode: A “language” helpful for describing the general structure of a program in a general way, allowing users of other languages to understand. It is not a precise coding language, but a description of the essential parts of an algorithm. It looks like this is meant to act as an outline for you to interpret and then code into your language of choice. The book uses Fortran and C.

Example given:
- Decare necessary variables and arrays
- Initialize variables
- Do the actual calculation
- Store the actual results

Fortran Snippet:

radioactive decay main program in Fortran

```
c Simulation of radioactive decay
c Program to accompany "Computational Physics" by N. Giordano/H. Nakanishi
      program decay                                    *
c     declare the arrays we will need
      double precision n_uranium(100), t(100)
c use subroutines to do the work
      call initialize(n_uranium,t,tau,dt,n)
      call calculate(n_uranium,t,tau,dt,n)
      call store(n_uranium,t,n)
      stop
      end
```

- I can somewhat see similarities to how we would do this in Python, but it looks like here it spells things out more. (Ex - literally typing out ‘call’)
- The majority of this chapter further explains the code in Fortran and C.

1.4 - Testing Your Program

- In this section, it asks that you check the results of your code. Does the output look reasonable? Does it match with any available exact results? The book specifically states that this should not be treated as a last-minute chore. Honestly that was the gist of the chapter.

1.5 - Numerical Considerations

- Must consider errors, one of which being a round-off error. Present when numbers are presented with a finite number of digits. Numerical solutions are inherently sensitive to round-off errors, though usually they aren’t a severe problem.

Radioactive Decay problem:

- Here, we treat time as a discrete variable. Differential equation -> Difference equation
- “Discretization” of time, space, or both, is a common practice. This does bring up two questions, “How do we know that the errors introduced by this discreteness are negligible?” and “How do we choose the value of such a step size for a calculation?
- Calculation should always be repeated using several values of that step size.
- Must evaluate results, especially if an exact numerical solution is not available.
- While the Euler method works for radioactive decay, it fails when oscillatory motion is involved. **There isn’t one best method for solving ordinary differential equations.**

1.6 - Programming Guidelines and Philosophy

Certain guidelines that are recommended:

- Program Structure - Organize major tasks
- Use Descriptive Names (something I need to incorporate for my functions)
- Use Comment Statements (Also something I need to do)
- Sacrifice (Almost) Everything for Clarity
- Take Time to Make Graphical Output as Clear as Possible (quantities, axes, labels)