NENG 685 Fall 2017
HW 2 Due Oct. 25, 2017

Name:

On homework:

- If you work with anyone else, document what you worked on together.
- Show your work.
- Always clearly label plots (axis labels, a title, and a legend if applicable).
- Homework should be done "by hand" (i.e. not with a numerical program such as MATLAB, Python, or Wolfram Alpha) unless otherwise specified. You may use a numerical program to check your work.
- If you use a numerical program to solve a problem, submit the associated code, input, and output (email submission is fine).

Do not write in the table to the right.

Problem	Points	Score
1	20	
2	10	
3	15	
4	20	
5	10	
6	25	
Total:	100	

1. (20 points) We often measure convergence by comparing one iteration to the previous iteration (rather than the solution, since we presumably don't know what it is). Imagine that you have software that gives the following solution vectors

$$x_{n-1} = \begin{pmatrix} 0.45 \\ 0.95 \\ 0.2 \\ -0.05 \\ 0.6 \end{pmatrix} \qquad x_n = \begin{pmatrix} 0.5 \\ 0.9 \\ 0.3 \\ -0.1 \\ 0.5 \end{pmatrix}$$

Calculate

   a. The absolute and relative error using the 1 norm
   b. The absolute and relative error using the 2 norm
   c. The absolute and relative error using the infinity norm
   d. What is *least* restrictive (that is, what would cause the code to converge *first*)?

Image that now $x_{n-1} = (0.49, 0.92, 0.4, -0.09, 0.51)^T$. Recalculate the convergence values for a-c above.

What do you observe? How does the *least* restrictive norm change? What does that mean about how you might select convergence criteria?

2. (10 points) You have a piece of software that takes mesh spacing as an input variable. Imagine that you have the following relative error values for each mesh spacing (h) / number of mesh cells (N cells):

h	N cells	rel err
1.0	8	8.44660179e-03
0.5	16	2.30286448e-03
0.1	80	9.84273963e-05
0.05	160	2.48043656e-05
0.01	800	9.98488163e-07

One of the ways we characterize method performance is the order of convergence. We'd like to know how the error changes as we change the resolution of our discretization, in this case, mesh spacing.

Using a log-log plot, plot relative error as a function of

   a. mesh spacing and
   b. cell count.

What is the functional relationship between error and mesh spacing / number of cells?

3. (15 points) Consider the following four equally-spaced points on the interval $[x_0, x_3]$:

$$x_j = x_0 + jh \qquad j = 0, 1, 2, 3 \qquad h = (x_3 - x_0)/3 \, .$$

Using the formula for the interpolating polynomial $P_3(x)$ you used in Question 1b of Homework 2, integrate $P_3(x)$ to derive the following Newton-Cotes formula, often referred to as **Simpson's three-eights rule**:

$$I(f(x)) \approx I_3(f(x)) = \frac{3h}{8}\Big[f(x_0) + 3f(x_0 + h) + 3f(x_0 + 2h) + f(x_0 + 3h)\Big]$$

Note that you can substitute h in for the xs in P_3 because we are explicitly stating that the points are equally spaced (i.e. $x = x_0 + 3th$, where $t \in [0, 1]$) $\therefore dx = 3hdt$, and x_0, x_1, and x_2 are modified accordingly).
For this problem, don't worry about an error term.

4. (20 points) Consider the following integral:

$$I = \int_2^4 \frac{x}{\sqrt{x^2 - 1}} dx \, .$$

   (a) (6 points) Compute the integral exactly by hand.

   (b) (8 points) Write a code that performs Composite Simpson's 3/8 rule to compute the integral. I advise something like
   `I = CompSimp38(a,b,n)`
   where `a` and `b` are the endpoints and `n` is the number of points to use in the integration (which must be divisible by 3!).

   (c) (6 points) Experimentally determine the rate of convergence as a function of `h`. Specify the order of convergence.

5. (10 points) For $n = 100$ we will use this tridiagonal system of equations

$$\begin{aligned} 2x_0 - x_1 &= 0 \\ -x_{j-1} + 2x_j - x_{j+1} &= j \, , \qquad j = 1, \ldots, n-2 \\ -x_{n-2} + 2x_{n-1} &= n - 1 \end{aligned}$$

in a few different ways. You may use Python, MATLAB or another package or code language of your choice. Note: `NumPy` and `SciPy` are libraries that can be imported into Python and would be useful for this assignment.

   (a) (2.5 points) Use built in Python or MATLAB commands to construct $\mathbf{A}$ and $\vec{b}$.

   (b) (2.5 points) What is the condition number of $\mathbf{A}$? What does this tell us? *(Hint: Use a Python built-in.)*

   (c) (2.5 points) Solve this problem by explicitly inverting $\mathbf{A}$ and multiplying $\vec{b}$.

(d) (2.5 points) Use `scipy.linalg.solve` (Python) or the backslash operator (MATLAB) to solve the system.

Plot both your explicit (c) and numerical (d) solutions on the same plot. Include axis labels, a title, and a legend.

6. (25 points) There are several fundamental alpha decay series (neptunium, uranium, thorium, and actinium). Here we'll look at the neptunium (which is the only one without the parent occurring naturally) as shown in the figure:

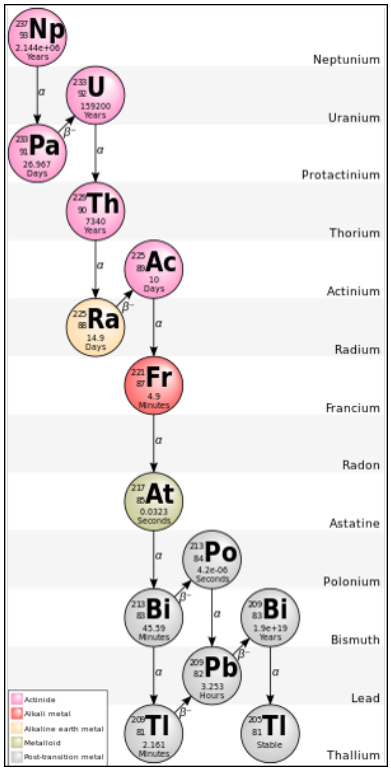

Figure 1: Neptunium Decay Series

This is a nice decay chain in the sense that there is only one non-linear term - the decay of ^{213}Bi, which can decay by β^- (97.8%) or α (2.2%). The population of each member, N, at any time t is given by the Bateman Equations:

$$\frac{dN_t(t)}{dt} = -\lambda_1 N_1(t)$$
$$\frac{dN_i(t)}{dt} = -\lambda_i N_i(t) + \lambda_{i-1} N_{i-1}(t)$$

which has the analytic solution of:

$$N_n(t) = \sum_{i=1}^{n} \left[N_i(0) \left(\prod_{j=1}^{n-1} \lambda_j \right) \left(\sum_{j=1}^{n} \left(\frac{e^{-\lambda_j t}}{\prod_{p=i,p\neq j}^{n} (\lambda_p - \lambda_j)} \right) \right) \right] \quad (1)$$

Find a solution to the system of ODEs for the neptunium series on a per atom basis. Plot the relative abundance of each isotope as a function of time spanning from 1 second to 1 day. Start with ^{221}Fr.

Hint(s): In the HW directory, there is a python dictionary containing all of the relevant nuclear data in a file named nuclearData.txt. You can copy this directly into your code and use it - in fact it is recommended. Also, there is no reason to code up an ODE solver. Instead use Python's ODEINT (demonstrated) in class.

Comment on what happens if you start higher in the decay series. What about if you carry the decay to longer times?

If you carry the decay far enough in time, you start to see oscillations in the population. Comment on the cause of these oscillations.

BONUS (5 points): submit your code by providing read/clone access to an online version control repository where your code is stored (e.g. github or bitbucket). If you don't know what that means and want to learn about it, come talk to me or check out resources here: `http://software-carpentry.org/lessons.html` For Windows, you want to setup the Git GUI. NOTE: You will not be able to do this from AFIT's systems.