PHYS 4023
Version 1

Homework Assignment #4: Root Finding, Interpolation, and Least Squares
Due Nov. 7th

Root Finding:

1. (Problem 5.5) Code up Newton's Method for our example function $f(x) = e^{x-\sqrt{x}} - x$ and reproduce figure 5.7 from the text. You should evaluate $f'(x)$ analytically.

2. (Problem 5.11) When using Newton's Method, it can sometimes be frustrating to start from different initial guesses but end up producing the same root over and over. A trick that can be used to suppress an already found root is to apply Newton's method not to $f(x)$ but to $u(x) = f(x)/(x-a)$, where a is the root you've already found and are no longer looking for. Implement this strategy for the case of our example function $f(x) = e^{x-\sqrt{x}} - x$, where you are suppressing the root $a = 1$. This means that you should always be converging to the other root, regardless of your initial guess.

3. (Problem 5.16) Explore the roots of the function:

$$f(x) = -x^5 + 4x^4 - 4x^3 + x^2e^x - 2x^2 - 4xe^x + 8x + 4xe^x - 8$$

using a method of your choice (Other than Newton which was used in the previous problems). Check your results using the *scipy* function $fsolve()$.

Interpolation:

1. (NB) Write your own function that performs linear interpolation. It should take in 3 arguments: an array of data x_i, an array of data y_i, and set of new values to evaluate the interpolate x_{new}. The function should return the interpolated values y_{new} for the specified x_{new} values. Use your function to interpolate the following data set: $x = [0, 1, 2, 3, 4]$ and $y = [2, 1, 3, 5, 1]$, with x_{new} being an array of 50 evenly spaced numbers in the range $[0, 4]$. Create a set of subplots that compare your interpolation function for this data to the $scipy.interp1d$ function (i.e., have data markers on both plots and show the interpolated results, one in each plot, as different color lines).

2. (NB) Construct a cubic spline interpolate for the data: $x = [1, 2, 3]$ and $y = f(x) = \ln(x)$. Perform this calculation as a matrix algebra problem, and compare that result with the built in cubic spline function in *scipy*. Evaluate the interpolates with 100 equally spaced data points in the range $[1, 3]$ and plot the results.

3. (NB) This problem examines data extrapolation. Given the following data: $x = [1, 2, 3, 4, 5]$ and $f(x) = [100.000, 25.000, 11.111, 6.250, 4.000]$, find $f(5.7)$. Compare the results of polynomial interpolation ($n = 1, 2, 3$) as well as cubic spline. Note the 'true' function used here is $f(x) = \frac{100}{x^2}$. Use this fact to establish which function performs best for this particular problem.

Least Squares:

1. (NB) Use the least squares method to determine best fit parameters for a linear model applied to the following data set: $x = [2.1, 6.2, 7.2, 10.5, 13.7]$ and $y = [2.90, 3.83, 5.98, 5.71, 7.74]$. Solve this problem two ways: First, use one of the linear algebra based methods discussed in class. Then use numpy's pre-built *lstsq* function to determine the best fit parameters and compare with your earlier calculation. Last, produce a plot comparing the linear model fit with the data. Make the plot with appropriate labels for the axes and add a legend to easily indicate which data is the model and which is the observed data.

2. (NB) Use the least squares method to fit a cubic polynomial to the following data set: $x = [-4, -2, -1, 0, 1, 3, 4, 6]$ and $y = [-35.1, 15.1, 15.9, 8.9, 0.1, 0.1, 21.1, 135]$. Find the values for the polynomial coefficients, and produce a plot comparing the model fit with the data similar to what you did in the last problem.

3. (NB) Implement the Newton-Gauss method to fit a function $f(t) = at/(b+t)$ to the following data: $t = [0.038, 0.194, 0.425, 0.626, 1.253, 2.500, 3.740]$ and $y = [0.050, 0.127, 0.094, 0.2122, 0.2729, 0.2665, 0.3317]$. Your code should iterate on the solution until the change in residual is less than 1%, and note how many iterations it takes for the solution to converge. Start with an initial guess where $a \leq 1.0$ and $b \leq 1.0$. Finally, plot your best fit function against the data to check the solution by eye, and calculate a χ^2 (not reduced) for the model assuming an error bar for each y-data of 3%.