Problem Set 1

Due Thursday Jan. 30, 12:30 pm

Comments

- Please submit as a PDF to Gradescope.
- Please generate the PDF using Quarto. Feel free to work in a Jupyter notebook and then convert to Quarto before rendering to PDF. If you'd like to use some other format, please check with me.
- Remember to note at the start of your document the names of any other students that you worked with on the problem set (or indicating you didn't work with anyone if that was the case) and then indicate in the text or in code comments any specific ideas or code you borrowed from another student or any online reference (including ChatGPT or the like).
- Your solution should not just be code - you should have text describing how you approached the problem and what the various steps were, though for simple problems, this can be quite short. Your code should have comments indicating what each function or block of code does, and for any lines of code or code constructs that may be hard to understand, a comment indicating what that code does.
- You do not need to (and should not) show exhaustive output, but in general you should show short examples of what your code does to demonstrate its functionality. The output should be produced as a result of the code chunks being run during the rendering process, not by copy-pasting of output from running the code separately (and definitely not as screenshots).

Problems

1. Write a function that implements a basic version of Newton's method for minimizing a function of one variable.
    a. Start with a simple implementation. Use an existing Julia package to implement the finite difference estimates for the gradient and Hessian. Think about the arguments and any defaults, as well as the type of the output. For the moment don't set the types of the input arguments. You can start by assuming the function takes only one argument and simply returns the value at which the function is minimized.
    b. Now consider returning richer output. Consider using a named tuple, a dictionary, or a struct. What seem like the advantages/disadvantages? Choose one and implement it.
    c. Set up an array and save the progression of values along the optimization path.
    d. Use a ChatBot to write the code. Compare it to your code and indicate strengths/weaknesses.
2. Define the matrix `A = [1:4  5:8  ones(Int64,4)]`.
    a. Predict the result of performing the following operations in Julia (before checking your

answers by running them). Note that the lines are meant to be run one-by-one in the same workspace, so when you change an array, this will affect the subsequent statements.

i. `x = A[3,:]`
ii. `B = A[2:2,1:3]`
3. `A[1,1] = 9 + A[2,3]`
4. `A[1:3,2:3] = [0 0; 0 0; 0 0]`
5. `A[1:2,2:3] = [1 3; 1 3]`
6. `y = A[1:4, 3]`
7. `A = [A [2 1 7; 7 4 5; ones(Int64,2,3)]]`
8. `C = A[[1,3],2]`
9. `D = A[[1,2,4],[1,3,4]]`