LAB # 06

02/18/2025

---

1 Introduction

This lab focuses on the techniques used when building Quasi-Newton root finding methods. While methods in class focused on building methods that avoid calculating the inverse of a matrix at each step, this lab will be centered around approximating the Jacobian and using that methodology to build variations of Newton's method.

2 Pre-Lab

As stated on the lab handout, a common technique for approximating derivatives are finite differences. Finite difference approximations use function evaluations. Two definitions can be found below.

Definition 2.1 (Forward Difference)

$$f'(s) \approx \frac{f(s+h) - f(s)}{h}$$

Definition 2.2 (Centered Difference)

$$f'(s) \approx \frac{f(s+h) - f(s-h)}{2h}$$

For the Pre-Lab, we are asked to consider the function $f(x) = \cos x$ and we are asked to find the derivative $f'(x)$ when $x = \pi/2$.

2.1 Approximations of the Derivative

This question asks us to approximate the derivative using h where h is a list of values beginning at 0.01 and halving for every consecutive element. There are nine elements in h. The approximations were calculated using python with and the code can be found in the GitHub repository. Using the both the Forward and Centered Difference methods, the derivative of $f(x)$ was found to be $f'(x) \approx -1$ when $x = \pi/2$. This is in agreement with the analytic solution.

2.2 Order of Approximations

This part of the Pre-Lab asks us to calculate the order of these methods. Python code was written to calculate the order using the following equation.

$$q = \frac{\log \frac{|x_{k+1} - x_k|}{|x_k - x_{k-1}|}}{\log \frac{|x_k - x_{k-1}|}{|x_{k-1} - x_{k-2}|}}$$

The code found that for each method, the convergence was linear with an order very close to 1.

3 Additional Quasi-Newton Methods

During lab, we will focus on constructing three different Quasi-Newton root-finding methods. The first being the *Slacker Newton* which is a take on the *Lazy Newton*. The remaining versions use the techniques from the Pre-Lab.

3.1 Building Slacker Newton

As we know, the Lazy Newton method only requires calculating the Jacobian once which reduces the likelihood of convergence. The Slacker Newton method also follows this idea but recalculates the Jacobian when it suspects that convergence is not happenning.

3.1.1

I chose to use the conditional check of the distance between the current and most recent iterate as my *slacker* condition. I chose to recalculate the Jacobian, J, when the distance, d, is less than $1.0 * 10^{-5}$.

3.1.2

Beginning with the *Lazy Newton* code from class, I was able to write my own *Slacker Newton* code which can be found in the GitHub repository.

3.1.3

In this part, we are asked to consider the function $\mathbf{f}(\mathbf{x})$ as shown below.

$$\mathbf{f}(\mathbf{x}) = \begin{bmatrix} 4x_1^2 + x_2^2 - 4 \\ x_1 + x_2 - \sin(x_1 - x_2) \end{bmatrix}$$

3.1.4

My *slacker* performed similar to that made by my lab partner.

3.2 Newton with Approximate Jacobian

I unfortunately ran out of time in lab so I was not able to complete this section.

3.3 Hybrid Newton

I unfortunately ran out of time in lab so I was not able to complete this section.

4 Deliverables

The code written for this lab and its associated exercises has been pushed to the Github repository under **Lab 06**. Additionally, the LaTeX code and rendering have also been pushed and submitted via Canvas.