MA 348 Numerical Analysis
Project 1

Kimberly George
David Jefts
Gabrielle Klein
Christopher McFall

28 February 2019

Introduction

Until this point we've learned how to find roots roots of nonlinear equations using either a bracketing or open methods, and how to solve systems of linear equations using Gaussian Elimination. This lab combines two previously learned techniques, Gaussian Elimination and the Newton Method of finding roots, to solve nonlinear systems of equations. The system at that is solved in this lab is shown in equation 1

$$\begin{cases} x^3 - 2y = 2 \\ x^3 - 5z^2 = -7 \\ yz^2 = 1 \end{cases} \tag{1}$$

Theory-Analysis

Newton's Method for linear system is an iterative method which uses an approximation of $\vec{f}(\vec{x}) = \vec{0}$. This approximation is derived from the Taylor series expansion of the function $f(x)$. The first terms of the Taylor series expansion at a point $f(x+\Delta x)$ is $f(x+\Delta x) \approx f(x) + f'(x)\Delta x$. Δx is solved for to figure out what the method's next approximation should be, yielding the equation for Newton's Method, $x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$. Similarly for a nonlinear system, the approximation for a function $\vec{f}(\vec{x})$ is $\vec{f}(\vec{x} + \Delta\vec{x}) \approx \vec{f}(\vec{x}) + D\vec{f}(\vec{x})\Delta\vec{x}$ where $D\vec{f}$ is the Jacobian of the function. The goal is to find the root of $\vec{f}(\vec{x})$, so the approximation function can be set to zero. The new equation can be rearranged solved for $\vec{x}$ by using Gaussian Elimination.

Starting with an initial guess based on an estimation of where the function is equal to 0, $\Delta\vec{x}$ is found and added to the previous value of $\vec{x}$. The method runs again with $\vec{x} + \Delta\vec{x}$ as its new $\vec{x}$, improving its estimate for $\vec{x}$ at each iteration. $D\vec{f}(\vec{x})$ in this equation is the Jacobian, which is a matrix of partial derivatives as shown below.

$$\begin{bmatrix} \frac{\partial f_1}{\partial x} & \frac{\partial f_1}{\partial y} & \frac{\partial f_1}{\partial z} \\ \frac{\partial f_2}{\partial x} & \frac{\partial f_2}{\partial y} & \frac{\partial f_2}{\partial z} \\ \frac{\partial f_3}{\partial x} & \frac{\partial f_3}{\partial y} & \frac{\partial f_3}{\partial z} \end{bmatrix} \tag{2}$$

Numerical Solution

The language used to solve this problem was Python. The program consisted of a loop which ran until the approximate error of each variable was within a user specified tolerance. The first step of the program received an initial guess for $\vec{x}$. Then it calculated $\vec{f}(\vec{x})$ which outputs a new vector of the same size. The fuction $\vec{f}$ is hardcoded to output a vector $< x^3 - 2y - 2, x^3 - 5z^2 + 7, yz^2 - 1 >$. The Jacobian is then calculated at the new value for $\vec{x}$. The Jacobian was also hardcoded as a Python function which accepts $\vec{x}$ and calculates Equation 2 at those values. Finally an augmented matrix is created by appending $-\vec{f}(\vec{x})$ to the calculated matrix to represent the equation $D\vec{f}(\vec{x})\Delta\vec{x} = -\vec{f}(\vec{x})$ in matrix form. Gaussian Elimination is then used to solve for $\Delta\vec{x}$.

Results and Discussion

The initial guess for $\vec{x}$ the algorithm was $\begin{bmatrix}1\\2\\3\end{bmatrix}$. The program calculated both $\vec{f}(\vec{x})$ and $D\vec{f}(\vec{x})$, which when put into its augmented matrix form was:

$$\left[\begin{array}{ccc|c} 3 & -2 & 0 & -5 \\ 3 & 0 & -30 & -37 \\ 0 & 9 & 12 & 17 \end{array}\right]$$

When this matrix was solved using Gaussian Elimination, $\Delta x = \begin{bmatrix}1.38\\-0.42\\1.09\end{bmatrix}$. This was then added to $\vec{x}$ to form the new $\vec{x} = \begin{bmatrix}2.38\\1.58\\1.91\end{bmatrix}$. This process was repeated until the approximate error of each entry in $\vec{x}$ is less than 0.0001. The individual steps of this process are tabulated below.

PUT TABLE OF VALUES HERE

As can be seen in the table, the method converged on a solution for $\vec{x}$ which made $\vec{f}(\vec{x})$ incredibly close to zero. Therefore the solution to this system of nonlinear equations is $\vec{x} \approx \begin{bmatrix}1.4422\\0.500\\1.4142\end{bmatrix}$. These values can be plugged into $\vec{f}(\vec{x})$ directly to show that they are a the correct solution.

Conclusion

The combined matrix methods used to solve non-linear matrices are very effective at solving matrices, however they have an extremely high operation cost. Each Gaussian Elimination is requires $O(n) = O(\frac{2}{3}n^3)$. The Jacobian steps had no computational operation cost since that step was hardcoded. The Newton's Method operational cost was only based on the number of times the program iterated until the desired tolerance was achieved. In total, this method had an operations cost of $O(n) = i \times (O(\frac{2}{3}n^3))$ where i is the number of iterations in the program. The LU Decomposition Method would have been unhelpful in this method because the Gaussian Elimination is applied to a different matrix each time instead of just a different b. In conclusion, the Gaussian Elimination and Newton's Matrix Methods are very effective at solving matrices and guarantee a solution but sacrifice operational power to do so.

Appendices

-