MAT 388/488 Project

Due Friday, June 11 at 5:45pm

This is a project based on the paper, A third-order Newton-type method to solve systems of nonlinear equations (2007). Your task is to use MatLab to implement the algorithm in Section 3 of the paper. The algorithm solves a system of equations by improving on the classical Newton iterative method. This algorithm can converge a lot faster than the classical method.

Summary of the Algorithm
The Newton iterative method for solving the system $\mathbf{F}(x) = \vec{0}$, where $\mathbf{F}\colon \mathbb{R}^n \to \mathbb{R}^n$ is given by

$$\vec{x}_{n+1} = \vec{x}_n - \mathbf{F}'(\vec{x}_n)^{-1}\mathbf{F}(\vec{x}_n), \tag{1}$$

where $\mathbf{F}'(\vec{x}_n)$ is the Jacobian matrix evaluated at $\vec{x} = \vec{x}_n$.
To solve the nonlinear system $\mathbf{F}(x) = \vec{0}$, the algorithm proposed in the paper is to use the following iteration method,

$$\vec{y}_n = \vec{x}_n - \mathbf{F}'(\vec{x}_n)^{-1}\mathbf{F}(\vec{x}_n), \tag{2}$$

$$\vec{x}_{n+1} = \vec{x}_n - \mathbf{F}'(\vec{x}_n)^{-1}\left(\mathbf{F}(\vec{x}_n) + \mathbf{F}(\vec{y}_n)\right). \tag{3}$$

This pair of equations is performed at each iteration. At iteration n, the current estimate of the solution is given by $\vec{x}_n$ and we use this estimate to update the best estimate, $\vec{x}_{n+1}$. We have used iterative method throughout this course to solve optimization problems. Equation (3) here corresponds to Equation (10) in Section 3 of the paper.

Example 1. Suppose we want to solve the following system of equations,

$$\begin{aligned} 10x_1 + \sin(x_1 + x_2) - 1 &= 0 \\ 8x_2 - \cos^2(x_3 - x_2) - 1 &= 0 \\ 12x_3 + \sin(x_3) - 1 &= 0 \end{aligned}$$

For this nonlinear system, $\mathbf{F}(x) = \vec{0}$, where $\mathbf{F} = (f_1, f_2, f_3)\colon \mathbb{R}^3 \to \mathbb{R}^3$, we can compute the Jacobian. In this example, $f_1 = 10x_1 + \sin(x_1 + x_2) - 1$ and the partial derivatives are

$$\frac{\partial f_1}{\partial x_1} = 10 + \cos(x_1 + x_2), \quad \frac{\partial f_1}{\partial x_1} = \cos(x_1 + x_2), \quad \frac{\partial f_1}{\partial x_3} = 0. \tag{4}$$

Next, we compute the partial derivatives of $f_2 = 8x_2 - \cos^2(x_3 - x_2) - 1$ and $f_3 = 12x_3 + \sin(x_3) - 1$ in order to obtain the Jacobian matrix $\mathbf{F}'$,

$$\mathbf{F}'(\vec{x}) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \frac{\partial f_1}{\partial x_3} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \frac{\partial f_2}{\partial x_3} \\ \frac{\partial f_3}{\partial x_1} & \frac{\partial f_3}{\partial x_2} & \frac{\partial f_3}{\partial x_3} \end{bmatrix}. \tag{5}$$

The three partial derivatives in Equation (4) indicate what the first row of the Jacobian matrix must be. It is worth emphasizing that the three variables x_1, x_2, x_3 are the three entries of $\vec{x}$. Keep in mind that in the algorithm, at each iteration, $\vec{x}$ gets updated, so the values of the three variables will change. Therefore, when we compute the Jacobian matrix in Equation (2), we always evaluate $\mathbf{F}'$ at the values of x_1, x_2, x_3 that correspond to $\vec{x}_n$.

Start with an initial vector $\vec{x}_0 = [x_1, x_2, x_3]$, with $x_1 = -100, x_2 = 500, x_3 = -300$. The algorithm converges at the end of 3 iterations, with

$$x_1 = 0.0689783,\ x_2 = 0.246442,\ x_3 = 0.0769289$$

Although the initial $\vec{x}_0$ is very far away, the algorithm converges to the solution in just 3 iterations.

Example 2. Find the values of x, y, z to minimize the following function.

$$f(x,y,z) = (x-1)^4 + (y-3)^4 + (z-5)^4 + (x-y)^6 + (y-z)^6 + (x-z)^6. \tag{6}$$

We compute the partial derivatives with respect to x, y, z, then set them to zero.

$$\begin{aligned}
\frac{\partial f}{\partial x} &= 4(x-1)^3 + 6(x-y)^5 + 6(x-z)^5 = 0 \\
\frac{\partial f}{\partial y} &= 4(y-3)^3 - 6(x-y)^5 + 6(y-z)^5 = 0 \\
\frac{\partial f}{\partial z} &= 4(z-5)^3 - 6(y-z)^5 - 6(x-z)^5 = 0
\end{aligned}$$

We have reduced the optimization problem, given by Equation (6), into solving a nonlinear system, $\mathbf{F}(x) = \vec{0}$, where $\mathbf{F} = (f_1, f_2, f_3)\colon \mathbb{R}^3 \to \mathbb{R}^3$. We now have the same setting as in Example 1. We want to solve for (x, y, z) so that $f_1 = 0, f_2 = 0, f_3 = 0$. Here, $f_1 = 4(x-1)^3 + 6(x-y)^5 + 6(x-z)^5$. We can compute the Jacobian. Among the 9 partial derivatives, 3 of them are shown below.

$$\mathbf{F}'(\vec{x}) = \begin{bmatrix} \frac{\partial f_1}{\partial x} & \frac{\partial f_1}{\partial y} & \frac{\partial f_1}{\partial z} \\ \frac{\partial f_2}{\partial x} & \frac{\partial f_2}{\partial y} & \frac{\partial f_2}{\partial z} \\ \frac{\partial f_3}{\partial x} & \frac{\partial f_3}{\partial y} & \frac{\partial f_3}{\partial z} \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1}{\partial x} & \frac{\partial f_1}{\partial y} & -30(x-z)^4 \\ -30(x-y)^4 & \frac{\partial f_2}{\partial y} & \frac{\partial f_2}{\partial z} \\ -30(x-z)^4 & \frac{\partial f_3}{\partial y} & \frac{\partial f_3}{\partial z} \end{bmatrix}.$$

Apply the same process in Example 1. We can solve this problem by the algorithm.

Example 3. Find the values of x_1, x_2 to minimize the following function.

$$f(x_1, x_2) = \frac{1}{4}\sum_{j=1}^{N} (y_j - x_1 t_j - x_2 u_j)^4. \tag{7}$$

This is a regression problem. We have N observations $(t_1, u_1, y_1), (t_2, u_2, y_2), (t_3, u_3, y_3), \dots (t_N, u_N, y_N)$. Suppose $N = 200$. This regression problem is often written as,

$$\min_{\vec{x} \in \mathbb{R}^2} \|\vec{y} - A\vec{x}\|_4 \tag{8}$$

where the data matrix A is 200×2 and the column vector y has 200 entries.
The numbers $t_1, t_2, t_3, \dots, t_N$ are the entries in the first column of A.
The numbers $u_1, u_2, u_3, \dots, t_N$ are the entries in the second column of A.
The numbers y_j in equation (7) are the entries in the vector $\vec{y}$ in equation (8). Do not confuse this with the variable $\vec{y}$ in the algorithm.
The norm $\|\cdot\|_4$ in equation (8) is an example of l_p norm $\|\cdot\|_p$. If we have $\|\cdot\|_2$, then we have the least-squares problem commonly encountered in linear regression.

To minimize the function in equation (7), we take partial derivatives and set them to zero.

$$\begin{aligned}\frac{\partial f}{\partial x_1} &= \textstyle\sum_{j=1}^{N} (y_j - x_1 t_j - x_2 u_j)^3 (-t_j) = 0 \\ \frac{\partial f}{\partial x_2} &= \textstyle\sum_{j=1}^{N} (y_j - x_1 t_j - x_2 u_j)^3 (-u_j) = 0.\end{aligned}$$

We are solving for x, y so that the pair of equations above are satisfied, i.e. $F_1 = 0, F_2 = 0$.

$$\mathbf{F}'(\vec{x}) = \begin{bmatrix} \frac{\partial F_1}{\partial x_1} & \frac{\partial F_1}{\partial x_2} \\ \\ \frac{\partial F_2}{\partial x_1} & \frac{\partial F_2}{\partial x_2} \end{bmatrix} \tag{9}$$

where the top left entry is

$$\frac{\partial F_1}{\partial x_1} = \sum_{j=1}^{N} 3(y_j - x_1 t_j - x_2 u_j)^2 \cdot (t_j)^2.$$

Each of the 4 partial derivatives in the Jacobian matrix is a summation. In MatLab, we can use a For Loop to compute each summation.

Suppose $y_j = \sqrt{3} t_j + 7 u_j + \epsilon_j$, where ϵ_j is noise. Here, ϵ_j represents a heavy corruption by noise. The algorithm converges to $x_1 = 1.982776,\ x_2 = 6.723010$ at the end of 6 iterations.

Example 4. Find the values of x_1, x_2 to minimize the following function.

$$f(x_1, x_2) = \frac{1}{2} \sum_{j=1}^{N} \left[y_j - \exp(x_1 t_j) - \exp(x_2 u_j)\right]^2. \tag{10}$$

This is a regression problem. We have N observations $(t_1, u_1, y_1), (t_2, u_2, y_2), (t_3, u_3, y_3), \dots (t_N, u_N, y_N)$. Take the partial derivatives and set them to zero.

$$\begin{aligned}\frac{\partial f}{\partial x_1} &= \textstyle\sum_{j=1}^{N} [y_j - \exp(x_1 t_j) - \exp(x_2 u_j)] (-\exp(x_1 t_j) t_j) = 0 \\ \frac{\partial f}{\partial x_2} &= \textstyle\sum_{j=1}^{N} [y_j - \exp(x_1 t_j) - \exp(x_2 u_j)] (-\exp(x_2 u_j) u_j) = 0.\end{aligned}$$

We have a pair of equations $F_1 = 0, F_2 = 0$. Apply the same reasoning in Example 3.

Problems for you to do

For Example 3 and Example 4, suppose we are given the input data: Amat is a matrix with $N = 200$ rows and 2 columns, which stores the values of t_j and u_j, bvect is a vector containing the values y_j; for $1 \leq j \leq 200$.

MatLab code that solves the problems in Example 1 and Example 3 are supplied to you. They illustrate how to implement the algorithm.

Task 1.
Solve the problem in Example 2. Use the MatLab code for Example 1, make the necessary changes; use that to help you with the problem in Example 2. Change the entries in the Jacobian matrix

```
F_prime_x
```

. For the initial vector $\vec{x}_0$, use $x_1 = -1, x_2 = 1, x_3 = -1$.

```
initial = [-1; 1; -1];
```

Run the algorithm for 10 iterations.
Write down the values of the three variables after iteration 1, iteration 2, and iteration 8.

Task 2.
Solve the problem in Example 4. A template for this problem, which sets up the data, is supplied to you. Use the MatLab code for Example 3, make the necessary changes to it; use that to help you with the problem in Example 4. You will need to compute the following:

```
sum1, sum2, sum3, sum4, sum5, sum6, sum7.
```

For the initial vector $\vec{x}_0$

```
initial = [0.75; 0.25];
```

Run the algorithm for 10 iterations.
Hand in your MatLab program which implements this algorithm.

Academic Integrity
It is important that you write your own MatLab code. This should not be code that you copy from other sources, such as code taken from the internet, or code written by other people.

MatLab
This is a mathematics project with a small amount of MatLab programming.
Suggestion: Use the templates of MatLab code that I sent the class, which includes the data. If you use these templates, it probably makes this project much easier for you. The important skills you need to do this project is the For loop.
The function

```
inv(Amat)
```

will compute the inverse of a matrix Amat.