Homework 4

Blake Hamilton

March 7, 2024

Question 1

We have the following nonlinear set of equations:

$$f(x, y) = 3x^2 - y^2 = 0,$$
$$g(x, y) = 3xy^2 - x^3 - 1 = 0.$$

(a) Iterating on the following system converges after 34 iterations to a precision of 10^{-10}. We know this because both the x and y values approach a constant instead of exploding and stop at 34 iterations.

$$\begin{bmatrix} x_{n+1} \\ y_{n+1} \end{bmatrix} = \begin{bmatrix} x_n \\ y_n \end{bmatrix} - \begin{bmatrix} 1/6 & 1/18 \\ 0 & 1/6 \end{bmatrix} \begin{bmatrix} f(x_n, y_n) \\ g(x_n, y_n) \end{bmatrix}, \quad n = 0, 1, 2, \ldots$$

$$\begin{bmatrix} x \\ y \end{bmatrix} \approx \begin{bmatrix} 0.5 \\ 0.866 \end{bmatrix}$$

(b) The above matrix is the inverse of the Jacobian, so the iteration is a Lazy Newton method. The motivation for using that matrix aims to do Newtons method in fewer steps by using the first matrix the entire time. We see this below:

$$J = \begin{bmatrix} 6x & -2y \\ 3y^2 - 3x^2 & 6xy \end{bmatrix} = \begin{bmatrix} 6 & -2 \\ 0 & 6 \end{bmatrix} \qquad \text{For } x = y = 1$$
$$J^{-1} = \begin{bmatrix} 1/6 & 1/18 \\ 0 & 1/6 \end{bmatrix}$$

(c) Using Newton's method, we converge in 18 iterations. This makes sense because we expect Newton's to converge faster (in iterations) than Lazy Newton's.

(d) From our numerical result, we can see that the solution is:

$$\begin{bmatrix} x \\ y \end{bmatrix} \approx \begin{bmatrix} 0.5 \\ 0.866 \end{bmatrix}$$

We can check this numerically:

$$f(x,y) = 3x^2 - y^2 = 0, \qquad g(x,y) = 3xy^2 - x^3 - 1 = 0$$

$$3x^2 = \frac{x^3+1}{3x} \qquad \text{Solve for } y^2$$

$$x = \frac{1}{2}$$

$$y = \sqrt{3(\frac{1}{2})^2} \approx 0.866 \qquad \text{Plug in solved } x$$

Question 2

(i)
```
Starting point:  [1 1]
[-1.81626407  0.8373678 ]
Newton: the error message reads: 0
Newton: took this many seconds: 0.0011318087577819823
Netwon: number of iterations is: 7
[nan nan]
Lazy Newton: the error message reads: 1
Lazy Newton: took this many seconds: 2.113225829601288
Lazy Newton: number of iterations is: 99999
[-1.81626407  0.8373678 ]
Broyden: the error message reads: 0
Broyden: took this many seconds: 0.0007866024971008301
Broyden: number of iterations is: 12
```

(ii)
```
Starting point:  [ 1 -1]
[ 1.00416874 -1.72963729]
Newton: the error message reads: 0
Newton: took this many seconds: 0.0007902026176452637
Netwon: number of iterations is: 5
[ 1.00416874 -1.72963729]
Lazy Newton: the error message reads: 0
Lazy Newton: took this many seconds: 0.0008583784103393555
Lazy Newton: number of iterations is: 36
[ 1.00416874 -1.72963729]
Broyden: the error message reads: 0
Broyden: took this many seconds: 0.0004805445671081543
Broyden: number of iterations is: 6
```

(iii) The Jacobian matrix used for Newton and Lazy Newton is singular, as shown below.

$$\begin{bmatrix} 0.5 \\ 0.866 \end{bmatrix}$$

Because of this, Newton and Lazy Newton both fail, which shows the significance of the Broyden method.

```
[ 1.00416874 -1.72963729]
Broyden: the error message reads: 0
Broyden: took this many seconds: 0.0007001757621765137
Broyden: number of iterations is: 6
```

Conclusion: Although number of iterations is something to consider when analyzing algorithms, the most important factor in application is runtime. When Newton converges, Newton was always faster than Lazy Newton and Broyden. This is just based on circumstance, as there are situations where Lazy Newton will run faster. In the second set of starting points, we saw that Lazy Newton does not converge at all, which makes sense because as with standard Newton, Lazy Newton does not guarantee convergence. In the third scenario, the Jacobian is singular, so Newton and Lazy Newton both fail. As Broyden does not use the Jacobian, it converges and is therefore the clear winner with those starting points. Overall, Newton performs the fastest but Broyden's covers the most starting points (will converge for more possible starting points).

Question 3

```
Newton
Approximate root: [-4.24079751e-17  1.00000000e-01  1.00000000e+00]
Time taken (average over 20 runs): 0.0007594108581542968 seconds

Steepest Descent
Approximate root: [4.78493147e-17 9.99994548e-02 9.99999951e-01]
Time taken (average over 20 runs): 0.0032444238662719727 seconds

Steepest Descent Newton Hybrid
Approximate root: [4.78493147e-17 1.00000000e-01 1.00000000e+00]
Time taken (average over 20 runs): 0.0021424412727355955 seconds
```

Using $[0.5, 0.5, 0.5]$ as the starting point, it was observed that Newton's method exhibited the fastest convergence, reaching the desired tolerance of 10^{-6} the fastest. Newton's method is known for its quadratic convergence, which allows it to rapidly refine the solution with each iteration, making it particularly effective for well-conditioned problems. The steepest descent method showed much slower convergence compared to Newton's method. The hybrid approach, combining steepest descent with Newton's method, performed better than the pure steepest descent but still fell short of the efficiency achieved by Newton's method alone. This makes sense, as the hybrid method starts converging much quicker once it gets to the Newtonian piece of the algorithm.

Question 4

We are given the following interpolation data with a degree 4 polynomial $p(x)$:

x_j	0	2	3	5	8
y_j	-125	-27	-8	0	27

1. To write $p(x)$ using the Lagrange polynomial basis, we know that we have the following equation for each Lagrange term $p(x) = f(x_0)L_{n,0}(x) + \ldots + f(x_n)L_{n,n}(x) = \sum_{k=0}^{n} f(x_k)L_{n,k}(x)$, so in our case we have $p_n(x) = -125l_0 - 27l_1 - 8l_2 + 27l_4$. For each $L_{n,k}$, $L_{n,k}(x) = \prod_{\substack{j=0 \\ j\neq i}}^{n} \frac{x-x_j}{x_i-x_j}$, so we will calculate each of our L terms below:

$$L_0(x) = \frac{(x-2)(x-3)(x-5)(x-8)}{(0-2)(0-3)(0-5)(0-8)} = \frac{x^4 - 18x^3 + 111x^2 - 278x + 240}{240}$$

$$L_1(x) = \frac{(x-0)(x-3)(x-5)(x-8)}{(2-0)(2-3)(2-5)(2-8)} = \frac{x^4 - 16x^3 + 79x^2 - 120x}{12}$$

$$L_2(x) = \frac{(x-0)(x-2)(x-5)(x-8)}{(3-0)(3-2)(3-5)(3-8)} = \frac{x^4 - 15x^3 + 66x^2 - 80x}{30}$$

$$L_3(x) = \frac{(x-0)(x-2)(x-3)(x-8)}{(5-0)(5-2)(5-3)(5-8)} = \frac{x^4 - 13x^3 + 46x^2 - 48x}{90}$$

$$L_4(x) = \frac{(x-0)(x-2)(x-3)(x-5)}{(8-0)(8-2)(8-3)(8-5)} = \frac{x^4 - 10x^3 + 31x^2 - 30x}{720}$$

Now, we plug in each L to the earlier mentioned equation and get the following results many terms cancel: $p_n(x) = -125l_0 - 27l_1 - 8l_2 + 27l_4 = x^3 - 15x^2 + 75x - 125 = (x-5)^3$. This confirms the information that this data was sampled from $f(x) = (x-5)^3$, giving an accurate $p(x)$.

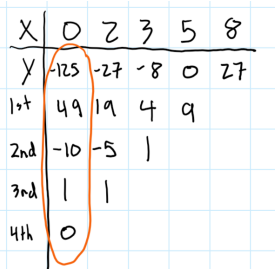
2. For a Newton basis, given a set of interpolation points $x_0, x_1, \ldots, x_n$, we have:

$$N_i(x) = \prod_{j=0}^{i-1}(x - x_j) \quad \text{for } i = 0, 1, \ldots, n$$

From this, we get the Newton polynomial:

$$\begin{aligned} P(x) = & f[x_0] \\ & + f[x_0, x_1](x - x_0) \\ & + f[x_0, x_1, x_2](x - x_0)(x - x_1) \\ & + \dots \\ & + f[x_0, x_1, \dots, x_n](x - x_0)(x - x_1)\dots(x - x_{n-1}) \end{aligned}$$

We can find the coefficients for Newton interpolation by divided differences using $f[x_0, x_1, \dots, x_m] = \frac{f[x_1, x_2, \dots, x_m] - f[x_0, x_1, \dots, x_{m-1}]}{x_m - x_0}$.



Now, we can plug in these coefficients to get the resulting polynomial: $p(x) = -125 + 49x - 10x(x-2) + x(x-2)(x-3) = x^3 - 15x^2 + 75x - 125 = (x-5)^3$, showing, as with Lagrange, our polynomial is as expected.