CS 71 Lab 2

Jessie Li

September 27, 2023

1

Error E_{n+1} after $n+1$ iterations (absolute value assumed):

$$E_{n+1} = p_{n+1} - p = g(p_n) - g(p) \tag{1}$$

Substitute the Taylor expansion of $g(p_n)$ around p into (1):

$$E_{n+1} = g'(p)E_n + \frac{1}{2}g''(p)E_n^2 + \frac{1}{6}g'''(p)E_n^3 + O(E_n^4) \tag{2}$$

Note that if $g'(p) = 0$ and $g''(p) = 0$, then

$$\lim_{n\to\infty} E_{n+1} \approx \frac{1}{6}g'''(p)E_n^3 \tag{3}$$

where $\alpha = 3$ is the order of convergence $\lambda = \frac{1}{6}g'''(p)$ is the asymptotic error constant

Therefore, given the equation

$$g(x) = x - \phi(x)f(x) - \psi(x)f(x) \tag{4}$$

we determine $\phi(x)$ and $\psi(x)$ by setting $g'(p) = 0$ and $g''(p) = 0$.

*Remember $f(p) = 0$ by definition

$$
\begin{aligned}
g'(x) &= 1 - \phi'(x)f(x) - \phi(x)f'(x) - \psi'(x)f(x)^2 - 2\psi(x)f(x)f'(x) \\
g''(x) &= -(\phi''(x)f(x) + 2\phi'(x)f'(x) + \phi(x)f''(x)) + \psi''(x)f(x)^2 \\
&\quad + 2\psi'(x)f(x)f'(x) + 2(\psi'(x)f(x)f'(x) + \psi(x)f'(x)^2 + \psi(x)f(x)f''(x))
\end{aligned}
$$

$$
\begin{aligned}
g'(p) &= 1 - \phi(p)f'(p) = 0 \\
&\implies \phi(x) = \frac{1}{f'(x)} \text{ and } \phi'(x) = \frac{-f''(x)}{2f'(x)^2} \\
g''(p) &= -\frac{f''(x)}{f'(x)} + 2\psi(x)f'(x)^2 = 0 \\
&\implies \psi(x) = \frac{f''(x)}{2f'(x)^3}
\end{aligned}
$$

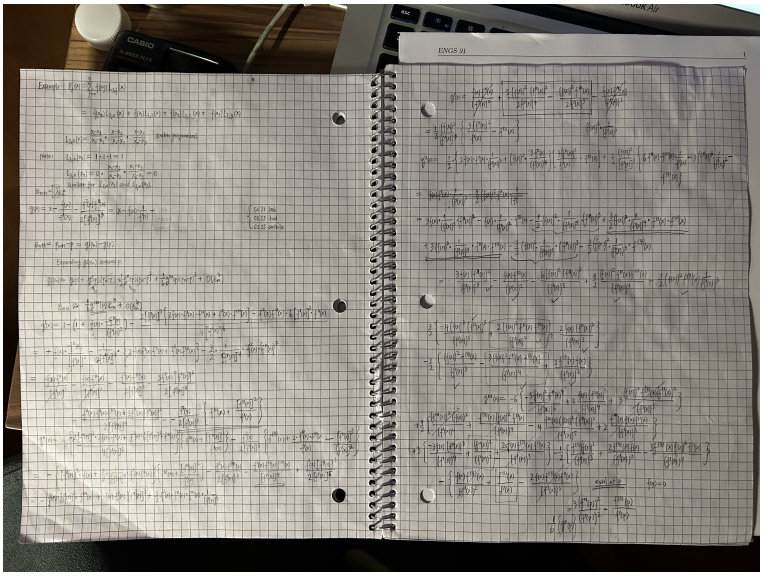
Plugging $\phi(x)$ and $\psi(x)$ into (4):

$$
g(x) = x - \frac{f(x)}{f'(x)} - \frac{f(x)^2 f''(x)}{2f'(x)^3} \tag{5}
$$

Corresponding fixed point iteration:

$$
x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} - \frac{f(x_n)^2 f''(x_n)}{2f'(x_n)^3} \tag{6}
$$

With $g(x)$, we can solve for $\lambda = \frac{1}{6}g'''(p)$ from (3) in terms of $f(p)$ and its derivatives. After extensive calculations...



...we arrive at:

$$\lambda = \frac{f''(p)^2}{2f'(p)^2} - \frac{f'''(p)}{6f'(p)} \tag{7}$$

2

General approach

- I defined a function for each method: Newton (A), Secant (B), Modified Newton (C), and Cubic Newton (D). All four shared common parameters:
  - f: function we want to find the root of
  - p0: initial approximation of root p
  - (Secant only) p1: second initial approximation of p
  - max_iter: maximum number of iterations
  - tol: tolerance
- Each function also returned:
  - n: number of iterations until distance between p_n and p_{n-1} < tolerance
  - p: approximation of p after n iterations
  - errors: array containing the error after each iteration
- By default, the maximum number of iterations was set to 100 and the tolerance was set to 2.2204×10^{-16}. This is the default tolerance for fzero, MATLAB's own root-finding function.

Pseudocode

Each method used a different fixed-point iteration scheme to determine the next approximation of p. These are below:

1. Newton's method

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

2. Secant method

$$x_{n+1} = x_n - \frac{f(x_n)(x_n - x_{n-1})}{f'(x_n) - f'(x_{n-1})}$$

3. Modified Newton's method

$$x_{n+1} = x_n - \frac{\mu(x_n)}{\mu'(x_n)}, \text{ where } \mu = \frac{f(x_n)}{f'(x_n)}$$

4. Cubic Newton's method

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} - \frac{f(x)^2 f''(x)}{2f'(x)^3}$$

Aside from calculating p_n differently, all four functions were implemented with the same general structure:

```
for n from 1 to max_iter
        calculate x[n+1] = g(x[n])
        calculate relative error[n+1]
                              = abs((x[n+1] - x[n])/x[n+1])
        if error[n+1] < tol
            stop
```

Error E_n and order of convergence α

- I used absolute relative error as my error metric
- I also used the definition of the order of convergence α from class:

$$\lim_{n\to\infty} \frac{E_n}{E_{n-1}^{\alpha}} = \lambda$$

- With at least two iterations, α can be calculated:

$$\begin{aligned} E_{n+1} &= \lambda E_n^{\alpha} \\ E_n &= \lambda E_{n-1}^{\alpha} \end{aligned}$$

$$\begin{aligned} E_{n+1}/E_n &= (E_n/E_{n-1})^{\alpha} \\ \log E_{n+1} - \log E_n &= \alpha \cdot (\log E_n - \log E_{n-1}) \end{aligned}$$

$$\implies \alpha = \frac{\log E_{n+1} - \log E_n}{\log E_n - \log E_{n-1}}$$

2.A

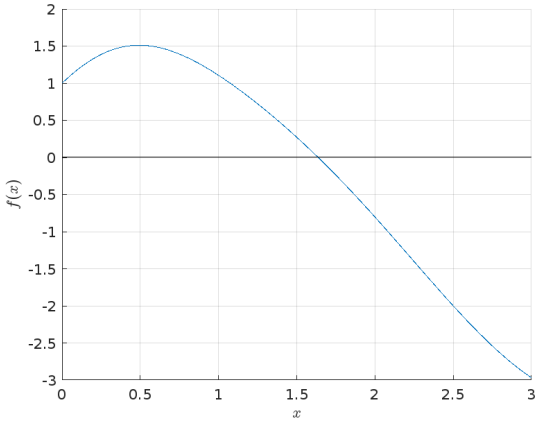


Figure 1: Graph of $f(x) = (x + \cos x)e^{-x^2} + x\cos x$

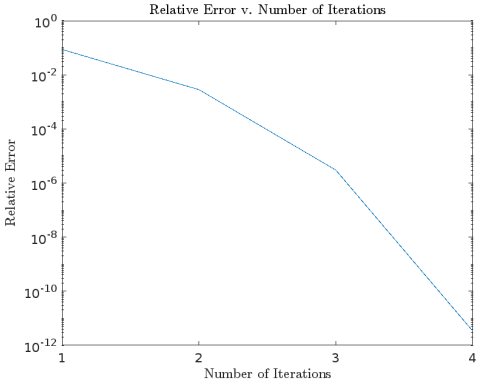
Observations

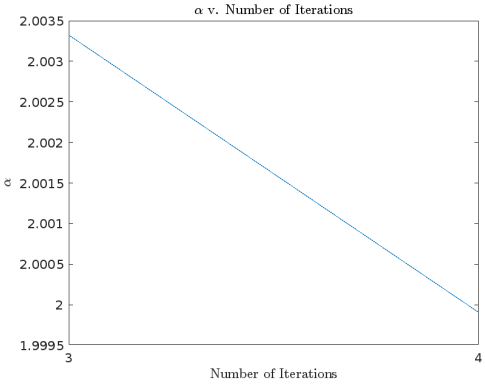
- Root p calculated with MATLAB's fzero is 1.636723
- $f'(p) \neq 0$, so we expect Newton's method to converge quadratically if we pick a starting point p_0 sufficiently close to p
- Based on the analysis from class, we also expect:
  - Secant to be superlinear ($\alpha = (1 + \sqrt{5})/2 \approx 1.618$)
  - Newton to be quadratic ($\alpha = 2$) with $\lambda = g''(p)/2 \approx 0.230168$
  - Modified Newton to be quadratic ($\alpha = 2$) with $\lambda = g''(p)/2 \approx 0.230168$
  - Cubic Newton to be cubic ($\alpha = 3$) with $\lambda = g'''(p)/6 \approx 0.121478$

Note

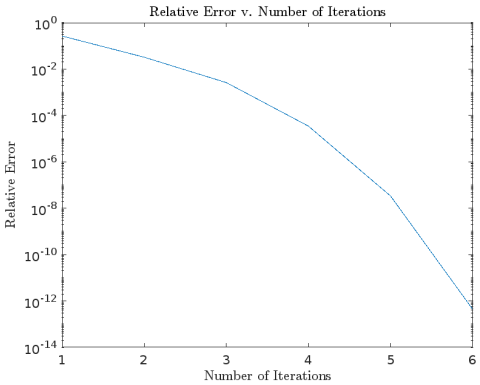
Most error graphs below show one less iteration than required for the error to become smaller than the default tolerance 2.2204×10^{-16} because MATLAB has limited precision. If the last error value is too small, MATLAB treats it like zero, which is outside the domain of log (the y-axis is on a log scale). For the same reason, the last α estimate may also not appear in the graph of α versus number of iterations because calculating α requires taking the logs of errors.

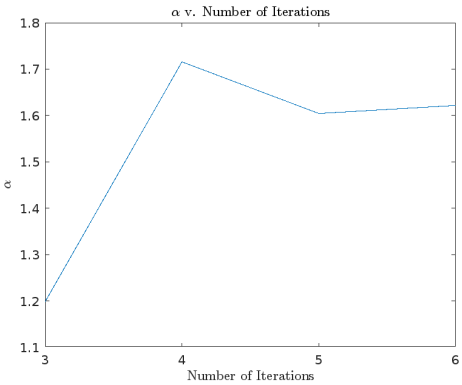
2.A.1 Newton's method



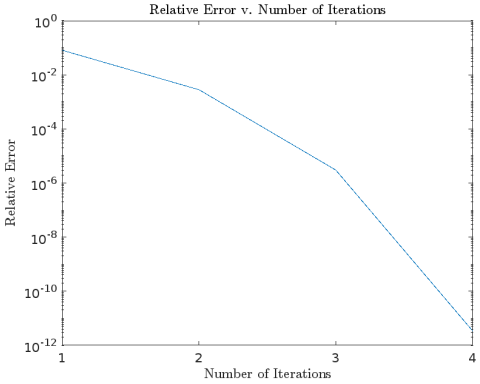


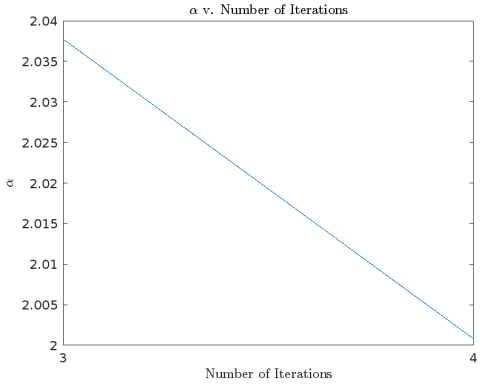
2.A.2 Secant method



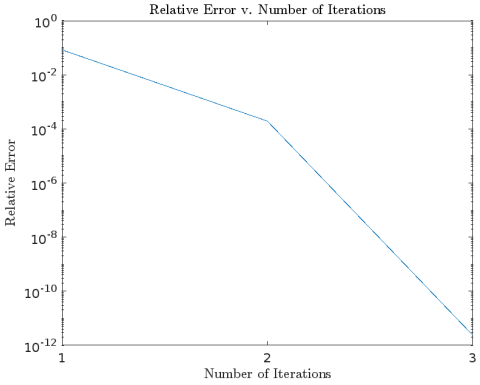


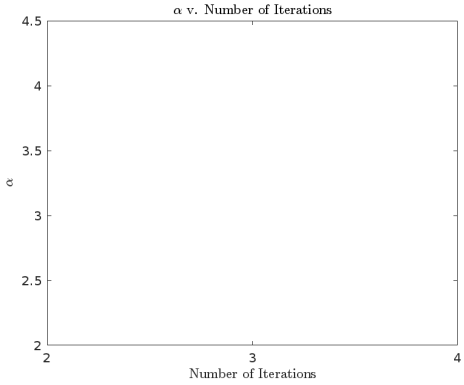
2.A.3 Modified Newton





2.A.4 Cubic Newton





Summary

Method	p_n	n	α_e	α_c	λ_e	λ_c
Newton	1.637	5	2	2.00	0.230	0.376
Secant	1.637	7	1.618	1.62	-	0.576
Modified Newton	1.637	5	2	2.00	0.230	0.381
Cubic Newton	1.637	4	3	3.00	0.122	3.35×10^{-4}

e = expected and c = calculated

- I hand-computed λ_c by taking $E_{n-1}/E_{n-2}^{\alpha_c}$. I used E_{n-1} and E_{n-2} instead of E_n and E_{n-1} because E_n was usually too close to zero

- Computed values of p and α are consistent with our expectations
- λ_c tends to deviate more from λ_e, but this could be because the sequences converged within a few iterations

2.B

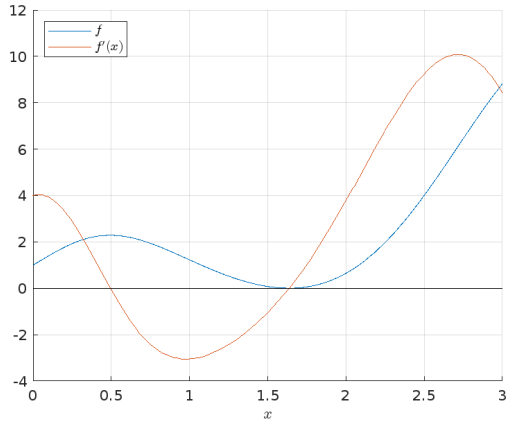
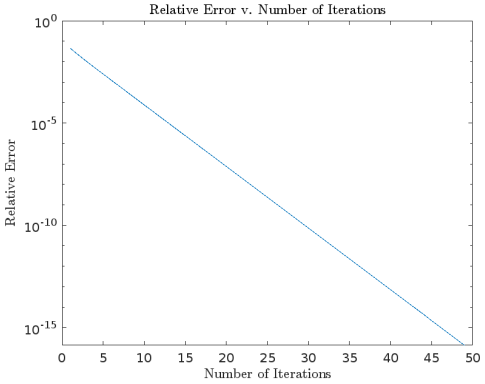


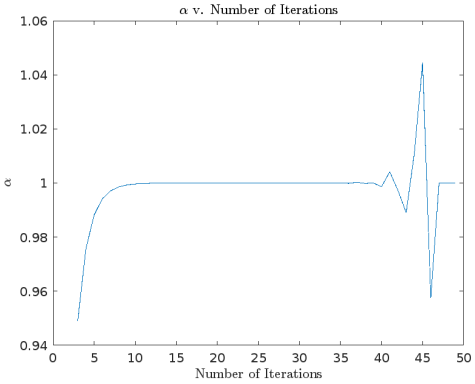
Figure 2: Graph of $f(x) = ((x + \cos x)e^{-x^2} + x\cos x)^2$ and its derivative, $f'(x)$

Observations

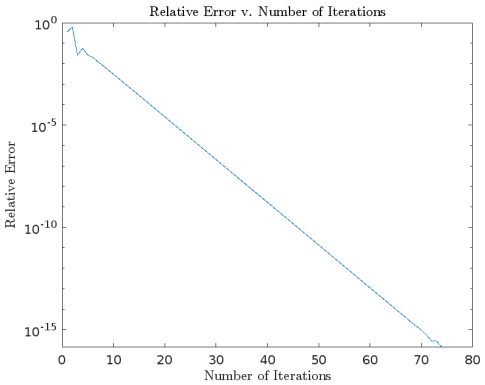
- Root p lies between (1, 2).
- $f'(p) = 0$, so Newton's method and cubic Newton's method should both converge linearly, since $g'(p)$ is non-zero
- Expect:
  - Newton's method to be linear ($\alpha = 1$) with $\lambda = g'(p) = 1/2$ (derived in class)
  - Cubic Newton's method to be linear ($\alpha = 1$) with $\lambda = g'(p) = 0.0.375000$
  - Secant method to be about linear (most likely $\alpha < 1.618$ and closer to 1 because Secant approximates Newton's method)
  - Modified Newton's method to be quadratic ($\alpha = 2$) with $\lambda = g''(p)/2 = 0.230168$

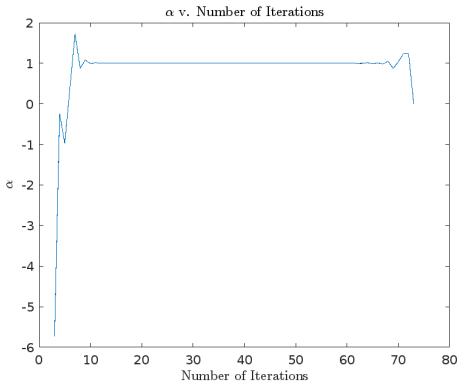
2.B.1 Newton's method



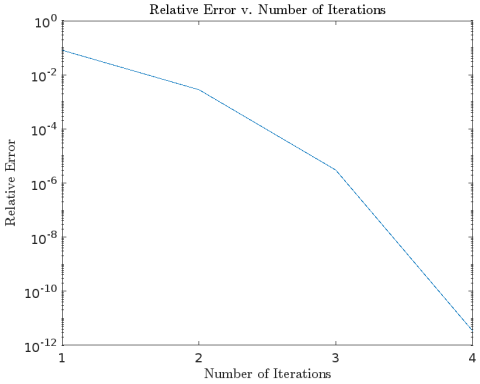


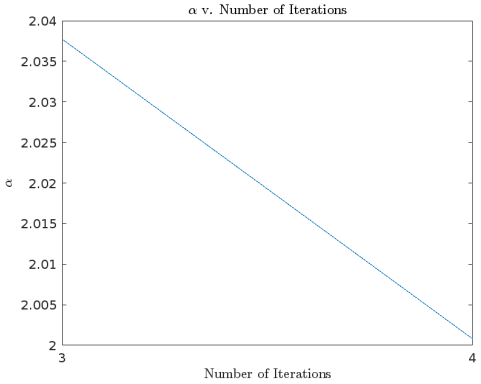
2.B.2 Secant method



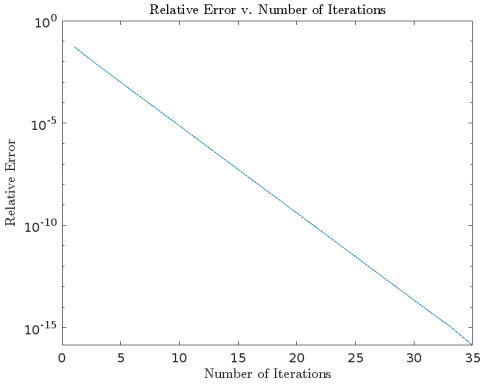


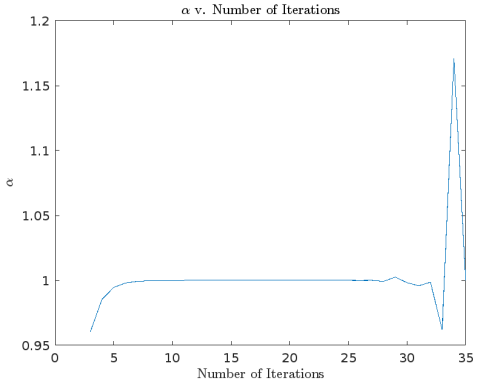
2.B.3 Modified Newton





2.B.4 Cubic Newton





2.B.5 Summary

Method	p_n	n	α_e	α_c	λ_e	λ_c
Newton	1.637	49	1	1.00	1/2	0.500
Secant	1.637	74	≈ 1	1.00*	-	0.500
Modified Newton	1.637	5	2	2.00	0.230	0.381
Cubic Newton	1.637	35	1	1.00**	0.375	0.333

e = expected and c = calculated

* The last values alpha were 0 and infinity, which were not representative of the overall graph, so I took the mode, $\alpha = 1$.

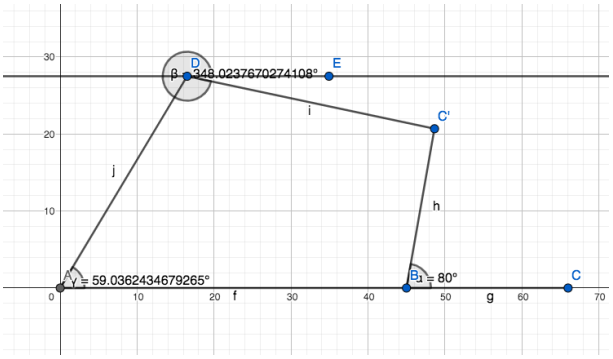
**Although the α printed by the program was roughly 1.17, the α occurring most often in the CSV file was 1.00.

- I hand-computed λ_c the same way I did in 2.A
- Computed values of p and α match expectations
- λ_c is fairly close to λ_e. Looking at the graphs of α versus number of iterations, deviations would not be surprising because of the inconsistencies observed within the last few (and the first few) iterations

3

Determining initial values with a GeoGebra sketch

I used the given values of r_1, r_2, r_3, r_4, and θ to sketch the figure in GeoGebra and determine good starting values for θ_2 and θ_3:



Observations:

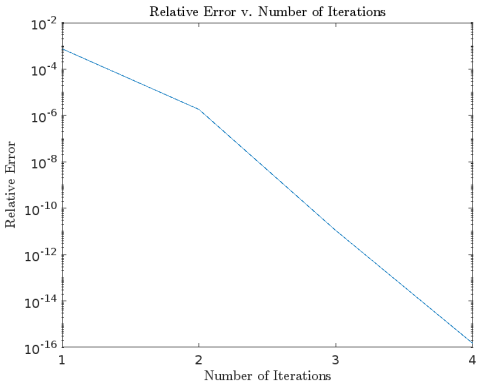
- $\theta_2 \approx 348°$
- $\theta_3 \approx 59°$
- $\theta_4 = \theta + 180°$

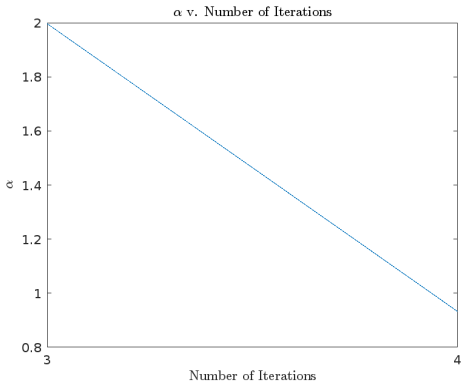
Notes

- I used Newton's fixed-point iteration scheme for nonlinear systems:

$$X_{n+1} = X_n - J^{-1}(x)F(x)$$

- Aside from extending Newton's method to a system, I used a similar approach and implementation as in Question 2
- I used the l2-norm in the computation of relative error
- I used MATLAB's in-built Gaussian elimination rref function on the augmented matrix [J F] to find $y = J^{-1}(x)F$, because calculating J^{-1} is typically much harder than using elimination to solve for y





Root found after 4 iterations of Newton's method, in degrees: $[\theta_2, \theta_3] \approx [59.25432, 348.07065]$. Order of convergence was roughly $1.996428 \approx 2$ (taking the second-to-last iteration), which indicates that $J(p)$ is not equal to the zero matrix. A quick MATLAB calculation of $J(p)$ (using the approximated p) indicates that this is indeed the case: $J(p) \approx [-27.5022, 6.8213; 16.3593, 32.2873]$.