

MATH 4640 Numerical Analysis - HW 3 Solutions

Austin Barton

April 20, 2025

Problem 1: Please see attached PDF with all written solutions for questions 1-5 and 7-8.

Problem 2: This problem accompanies the attached PDF which has a dedicated section for question 2. With the code in the following screenshots are the outputs for the code.

The screenshot shows a terminal window with the following content:

```
Sunday April 20, 21:47
ta

30 fn f1(x: f64) -> f64 {
29     x.exp() - 3. * x.powi(2)
28 }
27
26 fn grad_f1(x: f64) -> f64 {
25     x.exp() - 6. * x
24 }
23
22 fn f2(x: f64) -> f64 {
21     x - 1. - 0.2 * x.sin()
20 }
19
18 fn grad_f2(x: f64) -> f64 {
17     1. - 0.2 * x.cos()
16 }
15
14 fn g(x: f64) -> f64 {           █ function `g` is never used `#[warn(dead_code)]` on by default
13     x.powi(4) + 2. * x + 1.
12 }
11
10 fn grad_g(x: f64) -> f64 {
9     4. * x.powi(3) + 2.
8 }
7
6 fn grad_grad_g(x: f64) -> f64 {
5     12. * x.powi(2)
4 }
3
2 // x_{n+1} = x_n - (\grad f(x_n))^{-1} f(x_n)
1 fn newton_method(
0     f: impl Fn(f64) -> f64,
1         df: impl Fn(f64) -> f64,
2         x0: f64,
3         max_iter: u32,
4         delta: f64,
5         eps: f64,
6     ) -> Option<f64> {
7         let mut x: f64 = x0;
8         let mut v: f64 = f(x0);
9         let mut xi: f64;
10    }

Normal [p] hw4 [B] └─ W1 ┌─ ~/Documents/Academics/Math/Numerical/numerical_analysis_math4640/solutions/rust/hw4/src/q2.rs
Running 'target/debug/hw4 q2'
--MATH 4640, Numerical Analysis, Homework 4---

.....
Running solution to question 2 ...
Root of f1: x = 1.458962267530446
Root of f2: x = 1.0000000000000004
Minimizer of G(x): x = -0.7937005259840997
.....
End of Solution to question 2 :)
teddy@cabin:~/Documents/Academics/Math/Numerical/numerical_analysis_math4640/solutions/rust/hw4$
```

The terminal shows the execution of a Rust program named 'q2.rs' which performs numerical analysis tasks. It includes functions for f1, f2, g, grad_f1, grad_f2, grad_g, grad_grad_g, and newton_method. The program then runs a specific solution for question 2, displaying the roots of f1, f2, and the minimizer of G(x). The build output at the bottom shows the command 'Running 'target/debug/hw4 q2'' and the resulting binary file 'q2'.

Sunday April 20, 21:47

```
ta
```

```
26 // x_{n+1} = x_n - (\grad f(x_n))^{-1} f(x_n)
27 fn newton method(
28     f: impl Fn(f64) -> f64,
29     df: impl Fn(f64) -> f64,
30     x: f64,
31     mut mut_x: f64,
32     delta: f64,
33     eps: f64,
34 ) -> Option<f64> {
35     let mut x0: f64 = x;
36     let mut f0: f64 = f(x0);
37     let mut df0: f64 = df(x0);
38
39     if f0.abs() < eps {
40         return Some(x0); // root == x0
41     }
42
43     for _ in 0..m {
44         let dfx0 = df(x0);
45
46         if dfx0.abs() < le-15 {
47             println!("Derivative too small.");
48             return None;
49         }
50
51         x1 = x0 - (1. / dfx0) * v;
52         v = f(x1);
53
54         if (x1 - x0).abs() < delta || v.abs() < eps {
55             return Some(x1);
56         }
57         x0 = x1;
58     }
59
60     println!("Did not converge");
61     None
62 }
63
64 fn solve_q2() -> Result<(), Box

Normal:  $\sqrt{h4}$  B - A Wl + → /Documents/Academics/Math/Numerical/numerical_analysis_math_4640/solutions/rust/hw4/src/q2.rs    rust utf-8[unix] 1.82KiB 54:34



Running 'target/debug/hw4 q2'  
--MATH 4640, Numerical Analysis, Homework 4--



Running solution to question 2...  
Root of f1:  $x = -0.45962675369486$   
Root of f2:  $x = 1.195324038614666$   
Minimizer of G(x):  $x = -0.7937005259840997$



End of solution to question 2 :)



teddy@cabin rust/hw4 (hw4) >



[bum] B:rwim*



ImageGraphQ1.png Nov 16, 2024 Nov 16, 2024 Austin Teddy Barton 7.4 KiB



fregraphQ1.png Oct 21, 2024 Oct 21, 2024 Austin Teddy Barton 103 KB



good-data.jpg Mar 13, 2025 Mar 13, 2025 Austin Teddy Barton 37 KB



"cabin" 21:47 20-Apr-20


```

```
13     }
12
11     x1 = x0 - (1. / dfx0) * v;
10     v = f(x1);
9
8     if (x1 - x0).abs() < delta || v.abs() < eps {
7         return Some(x1);
6     }
5     x0 = x1;
4
3 } catch {
2     println!("Did not converge");
1     None
}
65 }

fn solve_q2() -> Result<(), Box
```

Problem 3: Please see attached PDF with all written solutions for questions 1-5 and 7-8.

Problem 4: Please see attached PDF with all written solutions for questions 1-5 and 7-8.

Problem 5: Please see attached PDF with all written solutions for questions 1-5 and 7-8.

Problem 6:

With the code in the following screenshots are the outputs for the code.

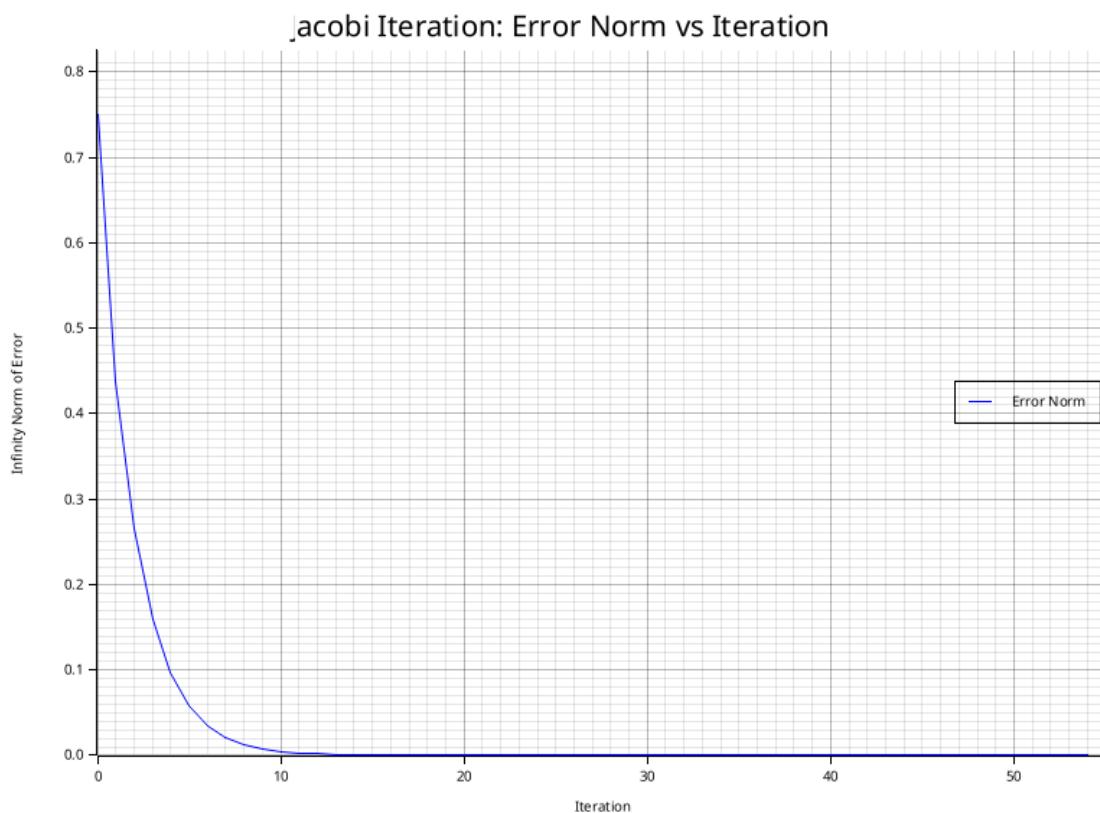
```

20 use plotters::prelude::*;
21
22 fn gauss_seidel_iteration(
23     a: &Vec<f64>,
24     b: &f64,
25     exact: &f64,
26     max_iter: usize,
27     tol: f64,
28 ) -> Result<(), Box<dyn std::error::Error>> {
29     let n = b.len();
30
31     let mut x_old = vec![0.0; n];
32     let mut x_new = vec![0.0; n];
33
34     let mut errors: Vec<usize, f64> = Vec::new();
35
36     for k in 0..max_iter {
37         for i in 0..n {
38             let mut sigma_1 = 0.0;
39             let mut sigma_2 = 0.0;
40
41             for j in 0..n {
42                 if j < i {
43                     sigma_1 += a[i][j] * x_new[j];
44                 }
45                 if j > i {
46                     sigma_2 += a[i][j] * x_old[j];
47                 }
48             }
49             x_new[i] = (b[i] - sigma_1 - sigma_2) / a[i][i];
50         }
51     }
52
53     Ok(())
54 }
55
56 fn jacobi_iteration(
57     a: &Vec<f64>,
58     b: &f64,
59     exact: &f64,
60     max_iter: usize,
61     tol: f64,
62 ) -> Result<(), Box<dyn std::error::Error>> {
63     let n = b.len();
64
65     let mut x_old = vec![0.0; n];
66     let mut x_new = vec![0.0; n];
67
68     let mut errors: Vec<usize, f64> = Vec::new();
69
70     for k in 0..max_iter {
71         for i in 0..n {
72             let mut sigma = 0.0;
73
74             for j in 0..n {
75                 if j != i {
76                     sigma += a[i][j] * x_old[j];
77                 }
78             }
79             x_new[i] = (b[i] - sigma) / a[i][i];
80         }
81     }
82
83     let error_norm = x_new
84         .iter()
85         .map(|x_i| x_i.abs())
86         .map(|(x_i, e_i)| (x_i - e_i).abs())
87         .fold(0.0, f64::max);
88
89     errors.push(k, error_norm);
90
91     Ok(())
92 }
93
94 fn main() {
95     let args: Vec<String> = std::env::args().collect();
96
97     if args.len() != 2 {
98         eprintln!("Usage: {} matrix_file", args[0]);
99         std::process::exit(1);
100    }
101
102    let matrix_file = &args[1];
103
104    let matrix = read_matrix(matrix_file);
105
106    let exact = matrix[0][0];
107
108    let mut tol = 1e-6;
109
110    let mut max_iter = 100;
111
112    let mut solver_type = "gauss_seidel";
113
114    for arg in args[2..] {
115        match arg.as_str() {
116            "jacobi" => solver_type = "jacobi",
117            "tol" >= arg[0..3] &gt;= "0.000000" &gt;= tol = arg[3..].parse().unwrap(),
118            "max_iter" >= arg[0..8] &gt;= max_iter = arg[8..].parse().unwrap(),
119            _ => eprintln!("Unknown argument: {}", arg),
120        }
121    }
122
123    let mut errors: Vec<f64> = Vec::new();
124
125    let mut start_time = std::time::SystemTime::now();
126
127    match solver_type {
128        "gauss_seidel" => {
129            let result = gauss_seidel_iteration(&matrix, &exact, &tol, max_iter);
130
131            if let Err(e) = result {
132                eprintln!("Error: {}", e);
133                std::process::exit(1);
134            }
135
136            let end_time = std::time::SystemTime::now();
137            let duration = end_time.duration_since(start_time).unwrap();
138
139            println!("Converged after {} iterations.", result.unwrap().0);
140            println!("Plot saved to plots/acobi_error_plot.png");
141            println!("Computed solution x:");
142            for i in 0..matrix[0].len() {
143                println!("{} ", matrix[0][i]);
144            }
145        }
146        "jacobi" => {
147            let result = jacobi_iteration(&matrix, &exact, &tol, max_iter);
148
149            if let Err(e) = result {
150                eprintln!("Error: {}", e);
151                std::process::exit(1);
152            }
153
154            let end_time = std::time::SystemTime::now();
155            let duration = end_time.duration_since(start_time).unwrap();
156
157            println!("Converged after {} iterations.", result.unwrap().0);
158            println!("Plot saved to plots/gauss_seidel_error_plot.png");
159            println!("Computed solution x:");
160            for i in 0..matrix[0].len() {
161                println!("{} ", matrix[0][i]);
162            }
163        }
164    }
165
166    println!("\n-----\nEnd of solution to question 6 :)\n");
167
168    teddy@cabin: ~$ ./hw4
169
170 [bum] 0:[tmux]*
```

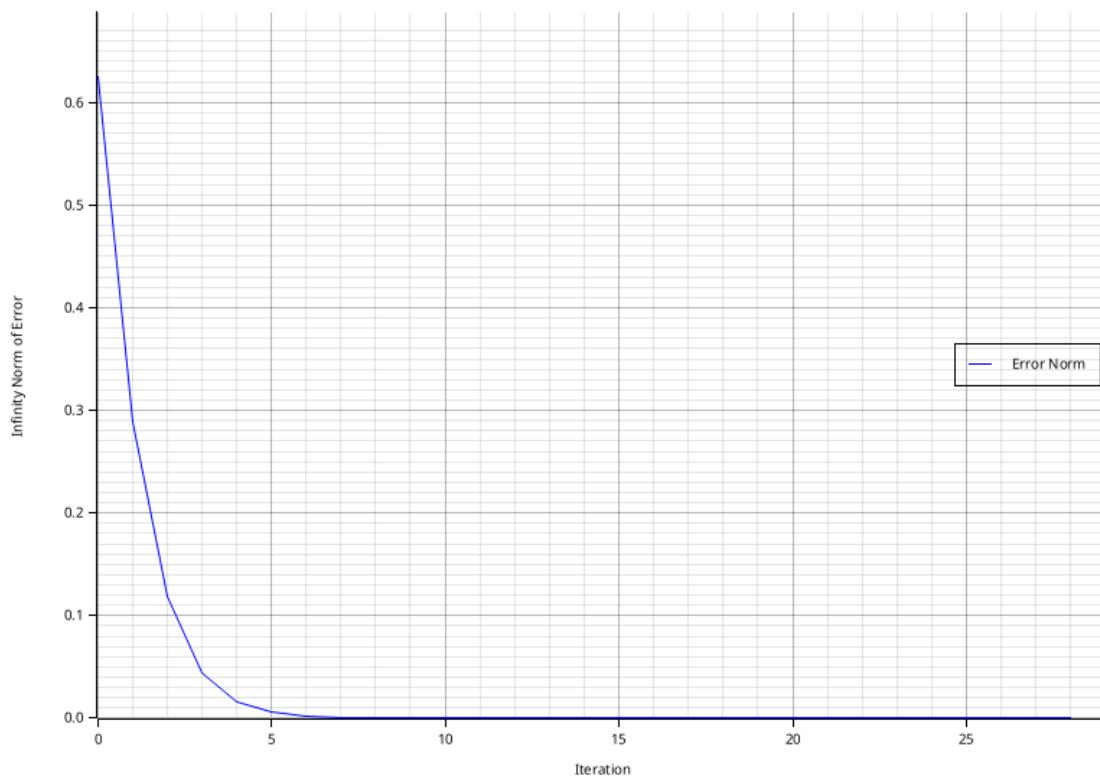
```

22 |     Ok(())
23 |
24 fn jacobi_iteration(
25     a: &Vec<f64>,
26     b: &f64,
27     exact: &f64,
28     max_iter: usize,
29     tol: f64,
30 ) -> Result<(), Box<dyn std::error::Error>> {
31     let n = b.len();
32
33     let mut x_old = vec![0.0; n];
34     let mut x_new = vec![0.0; n];
35
36     let mut errors: Vec<usize, f64> = Vec::new();
37
38     for k in 0..max_iter {
39         for i in 0..n {
40             let mut sigma = 0.0;
41
42             for j in 0..n {
43                 if j != i {
44                     sigma += a[i][j] * x_old[j];
45                 }
46             }
47             x_new[i] = (b[i] - sigma) / a[i][i];
48         }
49     }
50
51     let error_norm = x_new
52         .iter()
53         .map(|x_i| x_i.abs())
54         .map(|(x_i, e_i)| (x_i - e_i).abs())
55         .fold(0.0, f64::max);
56
57     errors.push(k, error_norm);
58
59     Ok(())
60 }
61
62 fn gauss_seidel_iteration(
63     a: &Vec<f64>,
64     b: &f64,
65     exact: &f64,
66     max_iter: usize,
67     tol: f64,
68 ) -> Result<(), Box<dyn std::error::Error>> {
69     let n = b.len();
70
71     let mut x_old = vec![0.0; n];
72     let mut x_new = vec![0.0; n];
73
74     let mut errors: Vec<usize, f64> = Vec::new();
75
76     for k in 0..max_iter {
77         for i in 0..n {
78             let mut sigma_1 = 0.0;
79             let mut sigma_2 = 0.0;
80
81             for j in 0..n {
82                 if j < i {
83                     sigma_1 += a[i][j] * x_new[j];
84                 }
85                 if j > i {
86                     sigma_2 += a[i][j] * x_old[j];
87                 }
88             }
89             x_new[i] = (b[i] - sigma_1 - sigma_2) / a[i][i];
90         }
91     }
92
93     let error_norm = x_new
94         .iter()
95         .map(|x_i| x_i.abs())
96         .map(|(x_i, e_i)| (x_i - e_i).abs())
97         .fold(0.0, f64::max);
98
99     errors.push(k, error_norm);
100
101    Ok(())
102 }
103
104 fn main() {
105     let args: Vec<String> = std::env::args().collect();
106
107     if args.len() != 2 {
108         eprintln!("Usage: {} matrix_file", args[0]);
109         std::process::exit(1);
110    }
111
112    let matrix_file = &args[1];
113
114    let matrix = read_matrix(matrix_file);
115
116    let exact = matrix[0][0];
117
118    let mut tol = 1e-6;
119
120    let mut max_iter = 100;
121
122    let mut solver_type = "jacobi";
123
124    for arg in args[2..] {
125        match arg.as_str() {
126            "gauss_seidel" => solver_type = "gauss_seidel",
127            "tol" >= arg[0..3] &gt;= "0.000000" &gt;= tol = arg[3..].parse().unwrap(),
128            "max_iter" >= arg[0..8] &gt;= max_iter = arg[8..].parse().unwrap(),
129            _ => eprintln!("Unknown argument: {}", arg),
130        }
131    }
132
133    let mut errors: Vec<f64> = Vec::new();
134
135    let mut start_time = std::time::SystemTime::now();
136
137    match solver_type {
138        "jacobi" => {
139            let result = jacobi_iteration(&matrix, &exact, &tol, max_iter);
140
141            if let Err(e) = result {
142                eprintln!("Error: {}", e);
143                std::process::exit(1);
144            }
145
146            let end_time = std::time::SystemTime::now();
147            let duration = end_time.duration_since(start_time).unwrap();
148
149            println!("Converged after {} iterations.", result.unwrap().0);
150            println!("Plot saved to plots/gauss_seidel_error_plot.png");
151            println!("Computed solution x:");
152            for i in 0..matrix[0].len() {
153                println!("{} ", matrix[0][i]);
154            }
155        }
156        "gauss_seidel" => {
157            let result = gauss_seidel_iteration(&matrix, &exact, &tol, max_iter);
158
159            if let Err(e) = result {
160                eprintln!("Error: {}", e);
161                std::process::exit(1);
162            }
163
164            let end_time = std::time::SystemTime::now();
165            let duration = end_time.duration_since(start_time).unwrap();
166
167            println!("Converged after {} iterations.", result.unwrap().0);
168            println!("Plot saved to plots/jacobi_error_plot.png");
169            println!("Computed solution x:");
170            for i in 0..matrix[0].len() {
171                println!("{} ", matrix[0][i]);
172            }
173        }
174    }
175
176    println!("\n-----\nEnd of solution to question 6 :)\n");
177
178    teddy@cabin: ~$ ./hw4
179
180 [bum] 0:nvim*
```

```
16 root.isPresent()?;
17 println!("Plot saved to plots/jacobi_error_plot.png");
18
19 println!("Computed solution x:");
20 for xi in X_new {
21     println!("({:.6})", xi);
22 }
23 Ok(())
24
25
26 fn solve_q6() -> Result<(), Box196
34         vec![0., 0., -1., 0., -1., 4.],
35     ];
36
37     let b: Vec<f64> = vec![2., 1., 2., 2., 1., 2.];
38
39     let exact: Vec<f64> = vec![1., 1., 1., 1., 1., 1.];
40
41     let tol: f64 = 1e-12;
42     let max_iter = 1000;
43
44     jacobi_iteration(&A, &b, &exact, max_iter, tol)?;
45
46     gauss_seidel_iteration(&A, &b, &exact, max_iter, tol)?;
47
48     Ok(())
49 }
50
51 pub fn main() {
52     if let Err(e) = solve_q6() {
53         eprintln!("Error: {}", e);
54         std::process::exit(1);
55     }
56 }
57 }
```



Gauss Seidel Iteration: Error Norm vs Iteration



Problem 7: Please see attached PDF with all written solutions for questions 1-5 and 6-7.

Problem 8: Please see attached PDF with all written solutions for questions 1-5 and 7-8.

1 Written Solutions

Q1 $p(x) = x^3 - 2x - 3$

$$p(0) = -3$$

$$p(1) = -4$$

$$p(2) = 1$$

There must be a root in $[0, 2]$ since $p(0)p(2) < 0$.

p is continuous on $[0, 2]$.

$$a = 0, b = 2$$

$$a_0 = a, b_0 = b$$

$$m_0 = \frac{a_0 + b_0}{2} = \frac{0+2}{2} = 1$$

$$p(m_0) = -4 \quad p(a_0)p(m_0) > 0, \quad p(m_0)p(b_0) < 0$$

$$a_1 = m_0, b_1 = b_0$$

$$m_1 = \frac{a_1 + b_1}{2} = \frac{1+2}{2} = \frac{3}{2}$$

$$p(m_1) = \left(\frac{3}{2}\right)^3 - 2\left(\frac{3}{2}\right) - 3 = \frac{27}{8} - 6 = \frac{27-48}{8} = -\frac{21}{8}$$

$$p(a_1)p(m_1) > 0, \quad p(m_1)p(b_1) < 0$$

$$a_2 = m_1, b_2 = b_1, \quad m_2 = \frac{\frac{3}{2} + 2}{2} = \frac{7}{4}$$

$$p(m_2) = \left(\frac{7}{4}\right)^3 - 2\left(\frac{7}{4}\right) - 3 = \frac{343}{64} - \frac{14}{4} = -\frac{73}{64}$$

$$p(a_2)p(m_2) > 0, \quad p(m_2)p(b_2) < 0$$

$$a_3 = m_2, \quad b_3 = b_2 \quad m_3 = \frac{\frac{7}{8} + 2}{2} = \frac{15}{8}$$

$$p(m_3) = \left(\frac{15}{8}\right)^3 - 2\left(\frac{15}{8}\right) - 3 = \frac{3375}{512} - \frac{30}{8} - 3$$

$$= \frac{3375}{512} - \frac{54}{8} = \frac{3375 - 3456}{512} = -\frac{81}{512}$$

$$p(a_3)p(m_3) > 0 \quad \text{so,}$$

$$a_4 = m_3, \quad b_4 = b_3 \quad m_4 = \frac{\frac{15}{8} + 2}{2} = \frac{31}{16}$$

$$p(m_4) = \left(\frac{31}{16}\right)^3 - 2\left(\frac{31}{16}\right) - 3 = \frac{29791}{4096} - \frac{62}{16} - \frac{48}{16}$$

$$= \frac{29791}{4096} - \frac{110}{16} = \frac{29791 - 28160}{4096} = \frac{1631}{4096}$$

$$p(a_4)p(m_4) < 0, \quad \text{so}$$

$$a_5 = a_4, \quad b_5 = m_4 \quad m_5 = \frac{\frac{15}{8} + \frac{31}{16}}{2} = \frac{61}{32}$$

$$m_5 = \frac{61}{32}$$

$$p(m_5) = \left(\frac{61}{32}\right)^3 - 2\left(\frac{61}{32}\right) - 3 = \frac{226981}{32768} - \frac{122}{32} - \frac{96}{32}$$

$$= \frac{226981}{32768} - \frac{218}{32} = \frac{226981 - 223232}{32768}$$

$$= \frac{3749}{32768}$$

$p(\text{las})p(\text{ms}) < 0$

$$a_6 = a_5, b_6 = m_5$$

$$m_6 = \frac{\frac{15}{8} + \frac{61}{32}}{2} = \frac{121}{64}$$

Exact $x^* = 1.89326919630450$

m_6 is within 3 significant figures of x^* .

Answer $\hat{x} = \underline{\underline{\frac{121}{64}}} \approx x^* \text{ (w/ 3 sig figs)}$

Analysis

$$\left[\frac{121}{64} = \underline{1.890625} \right]$$

A tolerance less or equal to $\frac{1}{16}$ would result in w being at least within 3 significant digits.

[2]

Q 2

(a) $f(x) = e^x - 3x^2$ $\frac{d^2}{dx^2} f(x) = \frac{d}{dx} f'(x) = \frac{d}{dx}(e^x - 6x) = e^x - 6$

So, $f \in C^2$.

Define: $c_n = x_n - x^*$

We know that since $f \in C^2$, for some neighborhood of x^* , if we start Newton's x_0 in that neighborhood, the convergence is quadratic.

(b) $f(x) = x - 1 - 0.2 \sin(x)$

$f'(x) = 1 - 0.2 \cos(x)$

$f''(x) = 0.2 \sin(x)$

So, $f \in C^2$.

Thus, Newton's method has quadratic convergence if started w/in some neighborhood of x^* .

(c) $G(x) = x^4 + 2x + 1$

Define $f(x) = \nabla G(x) = 4x^3 + 2x$

To minimize G , we find roots of f using Newton's Method

$f'(x) = 12x^2 + 2$

$f''(x) = 24x$

So, $f \in C^2$. Thus, Newton's method has quadratic convergence if started w/in some neighborhood of x^* .

G3 $x_{n+1} = 2 - (1+c)x_n + cx_n^3$, $\alpha = 1$ is a fixed point.

$$x_{n+1} = F(x_n)$$

F is differentiable and need $|F'(\alpha)| < 1$, ensure local convergence.

Now, $F(x) = 2 - (1+c)x + cx^3$

$$F'(x) = -1 - c + 3cx^2$$

Take the Taylor expansion of F about α :

$$\bullet x_{n+1} = F(x_n) = \alpha + F'(\alpha)e_n + \frac{1}{2}F''(\alpha)e_n^2 + \dots$$

meaning that...

$$\bullet e_{n+1} = x_{n+1} - \alpha = F'(\alpha)e_n + \frac{1}{2}F''(\alpha)e_n^2 + \dots$$

Now consider:

$$F'(\alpha) = F'(1) = -1 - c + 3c = -1 + 2c$$

We need $|F'(\alpha)| < 1 \Rightarrow -1 < F'(\alpha) < 1$

$$\Rightarrow -1 < -1 + 2c < 1 \Rightarrow 0 < 2c < 2 \\ \Rightarrow 0 < c < 1$$

S. $c \in (0, 1)$

But as we know this is for linear convergence.

To know what c grants quadratic convergence, we need the value of c such that $\underline{F'(\alpha) = 0}$

That is, $F'(1) = -1 + 2c = 0 \Rightarrow c = 1/2$

Answer: $c = 1/2$ will grant quadratic convergence. Why?

$$e_{n+1} = \frac{1}{2}F''(\alpha)e_n^2 + \dots \text{ in this case, since } F''(\alpha) = 0.$$

Q4

$$(a) \quad x_{n+1} = -16 + 6x_n + \frac{12}{x_n}, \quad x = 2$$

$$x_{n+1} = F(x_n), \quad F(2) = -16 + 12 + 6 = 2$$

$$F'(x) = 6 - \frac{12}{x^2} \text{ so } F \in C^1 \text{ near } x,$$

$$F'(2) = 6 - \frac{12}{4} = 6 - 3 = 3$$

$$|F'(2)| > 1 \Rightarrow \text{does not converge.}$$

$$(b) \quad x_{n+1} = \frac{2}{3}x_n + \frac{1}{x_n^2}, \quad x = 3^{1/3}$$

$$x_{n+1} = F(x_n), \quad F(3^{1/3}) = \frac{2}{3}3^{1/3} + \frac{1}{3^{2/3}} =$$

$$F(3^{1/3})3^{2/3} = \frac{2}{3}(3) + \frac{3^{2/3}}{3^{2/3}} = 2 + 1 = 3$$

$$F(3^{1/3})3^{2/3} = 3 \Rightarrow F(3^{1/3}) = 3^{1/3}$$

$$F'(x) = \frac{2}{3} - \frac{2}{x^3} \text{ so } F \in C^1 \text{ near } x.$$

$$F'(x) = \frac{2}{3} - \frac{2}{(3^{1/3})^3} = 0$$

$$|F'(x)| < 1, \text{ does converge.}$$

$$F''(x) = \frac{6}{x^4}, \quad F''(x) = \frac{6}{3^{4/3}} > 0$$

Since $F'(x) = 0$, $F''(x) \neq 0$ this iteration has quadratic convergence.

$$(c) \quad x_{n+1} = \frac{12}{1+x_n}, \quad x=3$$

$$x_{n+1} = F(x_n) \quad F(3) = \frac{12}{1+3} = \frac{12}{4} = 3$$

$$F'(x) = \frac{d}{dx} (12)(1+x)^{-1} = (-12)(1+x)^{-2} = -\frac{12}{(1+x)^2}$$

$F \in C^1$ near 3.

$$F'(x) = -\frac{12}{(1+x)^2} = -\frac{12}{16} = -\frac{3}{4}$$

$|F'(x)| < 1$, does converge.

Since $|F'(x)| > 0$, the linear term in the Taylor expansion

$$e_{n+1} = x_{n+1} - x = F'(x)e_n + \frac{1}{2} F''(\alpha)e_n^2 + \dots$$

dominates. So $|e_{n+1}| \leq 8|e_n| \quad \forall n \in \{0, 1\}$

Therefore, this iteration converges linearly.

□

Q5 ($n=3$)

$$A_3 = \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix}$$

$$L = \begin{bmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{32} & 1 \end{bmatrix} \quad U = \begin{bmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{bmatrix}$$

$$u_{11} = 2$$

$$u_{12} = -1$$

$$u_{13} = 0$$

$$U = \begin{bmatrix} 2 & -1 & 0 \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{bmatrix}$$

$$2l_{21} = -1 \Rightarrow l_{21} = -\frac{1}{2}$$

$$2l_{31} = 0 \Rightarrow l_{31} = 0$$

$$L = \begin{bmatrix} 1 & 0 & 0 \\ -\frac{1}{2} & 1 & 0 \\ 0 & l_{32} & 1 \end{bmatrix}$$

$$\frac{1}{2} + u_{22} = 2$$

$$u_{22} = \frac{3}{2}$$

$$U = \begin{bmatrix} 2 & -1 & 0 \\ 0 & \frac{3}{2} & u_{23} \\ 0 & 0 & u_{33} \end{bmatrix}$$

$$\frac{3}{2}l_{32} = -1$$

$$l_{32} = -\frac{2}{3}$$

$$L = \begin{bmatrix} 1 & 0 & 0 \\ -\frac{1}{2} & 1 & 0 \\ 0 & -\frac{2}{3} & 1 \end{bmatrix}$$

$$u_{23} = -1$$

$$\frac{2}{3} + u_{33} = 2$$

$$u_{33} = \frac{4}{3}$$

For $n=3$ $A_3 = LU$,

$$L = \begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ 0 & -2 & 1 \end{bmatrix} \quad U = \begin{bmatrix} 2 & -1 & 0 \\ 0 & \frac{3}{2} & -1 \\ 0 & 0 & \frac{1}{3} \end{bmatrix}$$

Notes from $n=3$ LU factorization

$$u_{11} = A_{11} = 2, u_{12} = A_{12} = -1, u_{13} = A_{13} = 0$$

$$l_{21} = -1/l_{11} = -1/u_{11}$$

$$l_{31} = 0 \quad l_{32} = -2/l_{11} = -1/u_{11}$$

$$u_{22} = \frac{3}{2} = 2 + l_{21} \quad l_{i+1} = -1/u_{ii+1-i}$$

$$u_{23} = -1$$

$$u_{33} = 1/l_{11} = 2 + l_{32} \quad u_{ii} = 2 + l_{ii-1}$$

$$u_{12} = u_{23} = -1 \quad u_{ii+1} = -1$$

Rule / Formula

$$u_{ii} = 2 + l_{ii-1}$$

$$u_{ii+1} = -1$$

$$l_{ii-1} = -1/u_{ii+1-i}$$

Checking for $n=4$

$$A = \begin{bmatrix} 2 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 2 \end{bmatrix}$$

Then our formula says

$$L = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -\frac{1}{2} & 1 & 0 & 0 \\ 0 & -\frac{2}{3} & 1 & 0 \\ 0 & 0 & -\frac{3}{4} & 1 \end{bmatrix} \quad U = \begin{bmatrix} 2 & -1 & 0 & 0 \\ 0 & \frac{3}{2} & -1 & 0 \\ 0 & 0 & \frac{4}{3} & -1 \\ 0 & 0 & 0 & \frac{5}{4} \end{bmatrix}$$

The pattern follows:

$$\left\{ \begin{array}{l} l_{ii+1} = -\frac{i-1}{i}, \quad l_{ii} = 1, \text{ otherwise } 0 \\ u_{ii+1} = -1, \quad u_{ii} = \frac{i+1}{i}, \text{ otherwise } 0 \end{array} \right.$$

Final answer for formula for

L, U where $A = LU$



Q7

$$\begin{bmatrix} 6 & 4 & 4 & 1 \\ 4 & 6 & 1 & 4 \\ 4 & 1 & 6 & 4 \\ 1 & 4 & 4 & 6 \end{bmatrix}$$

Let $v^{(0)} = \begin{bmatrix} 1/2 \\ 1/2 \\ 1/2 \\ 1/2 \end{bmatrix}$

$$k=1 : w = Av^{(0)} = \begin{bmatrix} 3+2+2+1/2 \\ 2+3+1/2+2 \\ 2+1/2+3+2 \\ 1/2+2+2+3 \end{bmatrix} = \begin{bmatrix} 15/2 \\ 15/2 \\ 15/2 \\ 15/2 \end{bmatrix}$$

$$v^{(1)} = \frac{w}{\|w\|} = \begin{bmatrix} 1/2 \\ 1/2 \\ 1/2 \\ 1/2 \end{bmatrix}$$

It just so happens that $v^{(0)}$ is eigenvector.

The associated eigenvalue is 15, since

$$\lambda^{(1)} = (v^{(1)})^T A v^{(1)} = 15, \text{ and}$$

We see that $v^{(0)} = v^{(1)}$, so $\lambda^{(1)}$ is the dominant eigenvalue corresponding to eigenvector $v^{(1)}$.

Q8

Q8

$$\begin{pmatrix} -2 & 1 & 1 \\ 1 & 3 & 1 \\ 1 & -1 & 3 \end{pmatrix}$$

I'm going to use Gershgorin Circle Theorem which states:

Every eigenvalue of a square matrix lies within at least one Gershgorin disk defined from its rows.

Each disk is centered at a_{ii} , with radius equal to the $\sum_{\substack{j=1 \\ j \neq i}}^3 |a_{ij}|$.

That is,

$$D_i = \left\{ z \in \mathbb{C} : |z - a_{ii}| \leq \sum_{\substack{j=1 \\ j \neq i}}^3 |a_{ij}| \right\}$$

- $a_{11} = -2$ and $\sum_{\substack{j=1 \\ j \neq 1}}^3 |a_{1j}| = |1| + |1| = 2$

This disk is $D_1 = \underline{\underline{[-4, 0]}}$.

- $a_{22} = 3$ and $\sum_{\substack{j=1 \\ j \neq 2}}^3 |a_{2j}| = |1| + |1| = 2$

This disk is $D_2 = \underline{\underline{[1, 5]}}$

- $a_{33} = 3$ and $\sum_{\substack{j=1 \\ j \neq 3}}^3 |a_{3j}| = |1| + |1| = 2$

This disk is $D_3 = \underline{\underline{[1, 5]}}$

Eigenvalues lie within $[-4, 0] \cup [1, 5]$

That is, $-4 \leq \lambda_i \leq 0$ or $1 \leq \lambda_i \leq 5$ for each λ_i