MCSC 6020G - Numerical Analysis
Assignment 2

Parikshit Bajpai

Note: Marcos Machado and I worked together on this assignment and therefore the codes and plots are common.

Question 1

To prove. *Every strictly column diagonally dominant matrix $A \in \mathbb{C}^{n\times n}$ is nonsingular.*

Proof. A matrix $A \in \mathbb{C}^{n\times n}$ is strictly column diagonally dominant if

$$|a_{j,j}| > \sum_{i=1, i\neq j}^{n} |a_{i,j}| \tag{1}$$

■

Question 2

(a) 2 norm of a square matrix

To prove. *For a matrix $A \in \mathbb{C}^{n\times n}$ given by the outer product $A = yx^T$, $\|A\|_2 = \|x\|_2\|y\|_2$.*

Proof. For a matrix $A \in \mathbb{C}^{n\times n}$,

$$\begin{aligned}
\|A\|_2 &= \max_{v\in\mathbb{C}^{n\times n}} \frac{\|Av\|_2}{\|v\|_2} && (2)\\
&= \max_{v\in\mathbb{C}^{n\times n}} \frac{\|yx^Tv\|_2}{\|v\|_2} \qquad (\because A = yx^T) && (3)\\
&= \max_{v\in\mathbb{C}^{n\times n}} \frac{|x^Tv|\|y\|_2}{\|v\|_2} && (4)\\
&= \max_{v\in\mathbb{C}^{n\times n}} \frac{|x^Tx|\|y\|_2}{\|x\|_2} \qquad (\text{Take } v = x) && (5)\\
&= \frac{\|x\|_2\|x\|_2\|y\|_2}{\|x\|_2} \qquad (\text{Cauchy-Schwarz, } |uv| \leq \|u\|_2\|v\|_2, \text{ equality holds if } u = v) && (6)\\
&= \|x\|_2\|y\|_2
\end{aligned}$$

■

(b) Condition number of matrix

To prove. *For invertible matrices $A \in \mathbb{C}^{n\times n}$ and $B \in \mathbb{C}^{n\times n}$, condition number $K_p(AB) \leq K_p(A)K_p(B)$, where $K_p(A) = \|A\|\|A^{-1}\|$.*

Proof. For matrix $AB \in \mathbb{C}^{n\times n}$,

$$K_p(AB) = \|AB\|_p\|(AB)^{-1}\|_p \tag{7}$$
$$= \|AB\|_p\|B^{-1}A^{-1}\|_p \qquad \left(\because (AB)^{-1} = B^{-1}A^{-1}\right) \tag{8}$$
$$\leq \|A\|_p\|B\|_p\|B^{-1}\|_p\|A^{-1}\|_p \qquad (\because \|AB\|_p \leq \|A\|_p\|B\|_p) \tag{9}$$
$$\leq \left(\|A\|_p\|A^{-1}\|_p\right)\left(\|B\|_p\|B^{-1}\|_p\right) \tag{10}$$
$$\leq K_p(A)K_p(B) \qquad \blacksquare$$

Question 3

(a) Monotonically decreasing convex function

To prove. *Function f is a monotonically decreasing convex function on $[0,\infty)$, where f is of the following form*

$$f(x) = \sum_{i=1}^{N} \frac{a_i^2}{(b_i^2 + x)^2} \qquad \textit{For } a_i,\, b_i \in \mathbb{R} \tag{11}$$

Proof.

$$\frac{df}{dx} = -2\sum_{i=1}^{N} \frac{a_i^2}{(b_i^2 + x)^3} \leq 0 \qquad \forall x \in [0,\infty) \tag{12}$$

$$\frac{d^2f}{dx^2} = 6\sum_{i=1}^{N} \frac{a_i^2}{(b_i^2 + x)^4} \geq 0 \qquad \forall x \in [0,\infty) \tag{13}$$

Since the first derivative of the function is negative in the given domain, the function is monotonically decreasing and since the second derivative is positive in the domain, the function is convex. ■

(b) Newton's method

Newton's method was implemented to find the roots of $f(x) = \delta$, where $0 < \delta < f(0)$. Since the convergence of Newton's method depends on the initial guess x_0, a number of "manufactured" solutions were tested for different combinations of a_i, b_i and N. Using the manufactured solutions, the implementation was verified to ensure that the algorithm works when provided with a suitable initial estimate. The different combinations used are listed in table 1 and figure 1 shows the residuals versus the number of Newton iterations. The plots show that the Newton method converges almost quadratically once it gets close to the real root. However, if the initial guess x_0 is not sufficiently close to the real root, Newton method can often converge at a very slow rate or not converge at all. Since the manufactured solutions were provided with initial guesses close enough to the real roots, non-convergence was not observed. However, in a number of cases, the method showed a very slow initial progress and then once it got close enough to the final estimate the rate of convergence became close to quadratic.

Newton's method was then tested for randomly generated a_i and b_i. For these tests, the δ was selected randomly between 0 and $f(0)$ and $x_0 = 0$ was used as initial estimate. However, $x_0 = 0$ is a bad estimate in most cases. Smaller the minimum a_i, steeper is the shape of the function and a Newton step from $x_0 = 0$ can result in divergence in a very small number of iterations. Also as the minimum b_i gets smaller, the rightmost vertical asymptote gets closer to the estimate leading to convergence issues.

(c) Newton's method with rational approximation

The function $f(x)$ can be better approximated through a rational function $g(x)$ defined as follows:

$$g(x) = \frac{A}{B + x} \tag{14}$$

Table 1: Parameters a_i, b_i and N used in the manufactured functions. δ and x_0 are also shown.

Parameters	**Case I**	**Case II**	**Case III**	**Case IV**	**Case V**	**Case VI**	**Case VII**	**Case VIII**	**Case IX**	**Case X**
N	1	1	2	2	3	3	4	4	5	5
a_1	5	7	1	2	1	1	1	1	1	1
b_1	3	3	1	7	1	2	3	1	3	1
a_2	-	-	2	3	5	2	3	2	2	1
b_2	-	-	3	5	3	1	1	2	1	1
a_3	-	-	-	-	2	1	1	3	2	1
b_3	-	-	-	-	2	1	1	3	1	1
a_4	-	-	-	-	-	-	2	4	2	1
b_4	-	-	-	-	-	-	2	4	2	1
a_5	-	-	-	-	-	-	-	-	1	1
b_5	-	-	-	-	-	-	-	-	1	1
δ	0.25	0.5	1.0	0.015	1.0	5.0	10.0	1.0	1.0	1.0
x_0	5.0	0.0	0.0	20.0	0.0	0.5	0.0	1.0	0.0	0.0

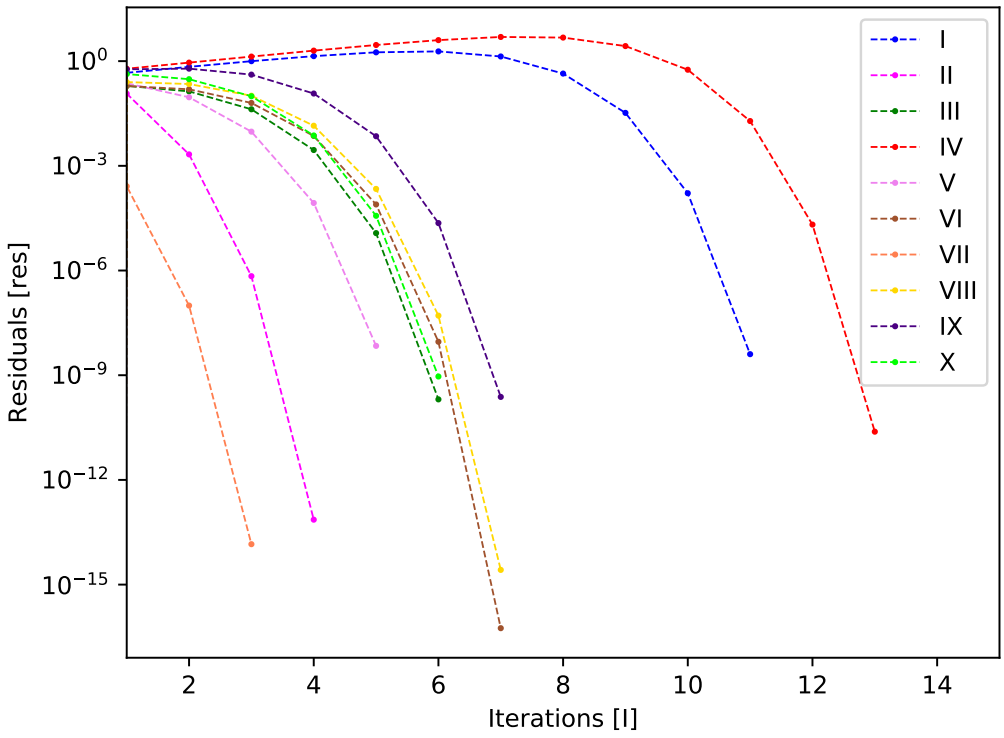


Figure 1: Residual versus number of Newton iterations for the manufactured functions using linear approximation.

Parameters A and B for iteration $n+1$ can be computed from the value of the function and its derivative at iteration n. If x_n is the estimate of the root at iteration n, then the parameters A and B can be estimated from the following equations:

$$g(x_n) = f(x_n) \tag{15}$$
$$g'(x_n) = f'(x_n) \tag{16}$$
$$\tag{17}$$

Solving the above equations for A_n and B_n gives:

$$A_n = \frac{\left(\sum_{i=1}^{N} \frac{a_i^2}{(b_i^2+x_n)^2}\right)^2}{2\left(\sum_{i=1}^{N} \frac{a_i^2}{(b_i^2+x_n)^3}\right)} \tag{18}$$

$$B_n = \frac{\left(\sum_{i=1}^{N} \frac{a_i^2}{(b_i^2+x_n)^2}\right)}{2\left(\sum_{i=1}^{N} \frac{a_i^2}{(b_i^2+x_n)^3}\right)} - x_n \tag{19}$$

where the subscript n highlights the fact that A and B are calculated at iteration n and change at each iteration.

The estimate of root at iteration $n+1$ is then the following:

$$x_{n+1} = \frac{A_n}{\delta} - B_n \tag{20}$$

(d) Implementation of Newton's method with rational approximation

Newton's method with rational approximation was implemented and tested for the same "manufactured" functions as the Newton's method with linear approximation using the same *delta* and initial guess x_0. This allowed a fair comparison of the two methods and also served as a test to verify the implementation since the results could be compared with the a-priori known results (from solving manufactured functions). Rational approximation shows much better convergence rate when compared to the linear approximation since it approximates the original function much closer than the linear approximation. Also, the method converges in less than ten iterations as compared to the linear approximation which can take thousands of iterations to converge depending on the values of a_i and b_i. Figure 2 shows the residuals versus the number of Newton iterations for the rational function case.

Like the linear approximation, the convergence of rational approximation also depends on the initial guess and the values of parameters a_i and b_i. Since the minimum a_i and minimum b_i are the dominant factors affecting the shape of the function, the asymptotes and subsequently the root, the following initial guess approximation can theoretically be used:

$$x_0 = \frac{\min_i(a_i)^2}{\delta} - \min_i(b_i)^2 \tag{21}$$

However, the method did not work when implemented and has therefore not been used in the codes. The method was also tested for randomly generated a_i and b_i. For these tests, the δ was selected randomly between 0 and $f(0)$ and $x_0 = 0$ was used as initial estimate. $x_0 = 0$ is still a bad initial guess and while the rational approximation resulted in convergence more often than the linear approximation, it did fail to converge numerous times. However, even when it failed to converge, the residuals and errors were still less than the linear approximation. A comparison of the two methods has been shown in figures 3 and 3.

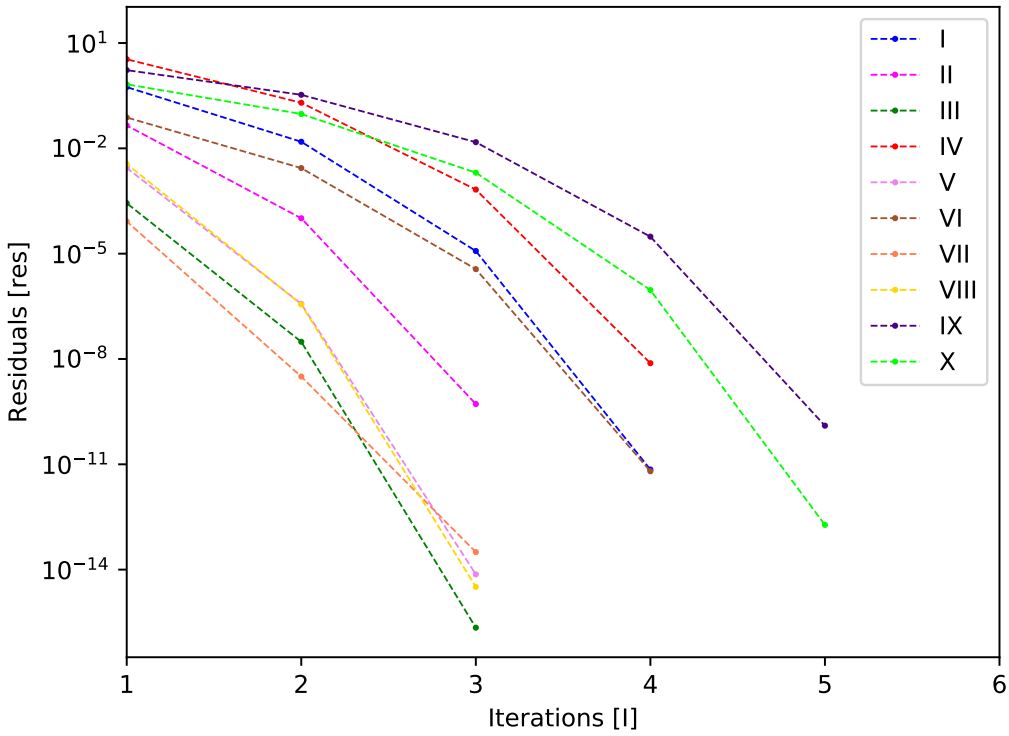
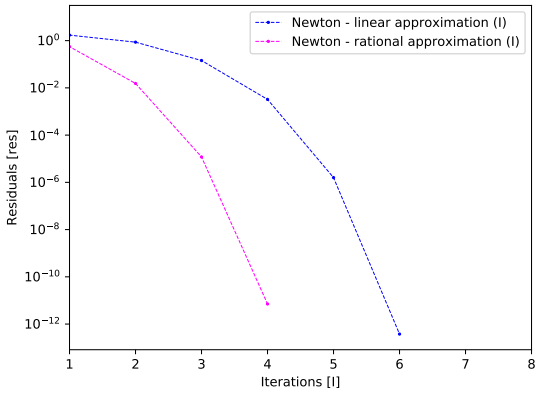
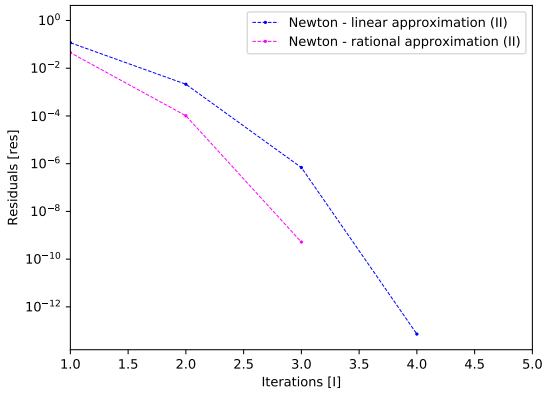


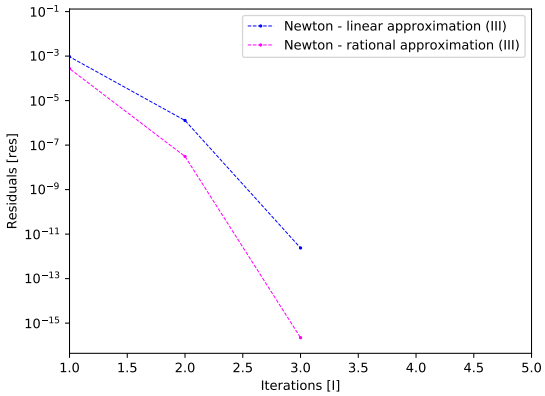
Figure 2: Residual versus number of Newton iterations for the manufactured functions using rational approximation.



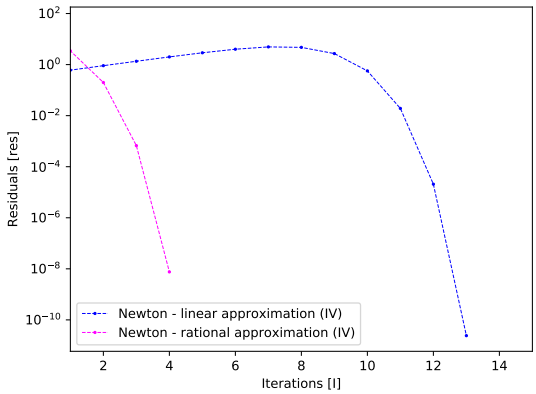
(a) Case 1



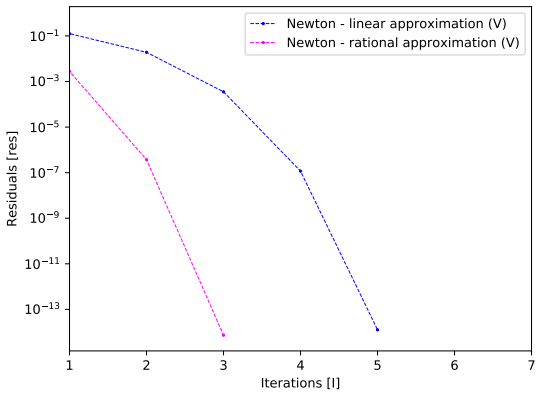
(b) Case 2



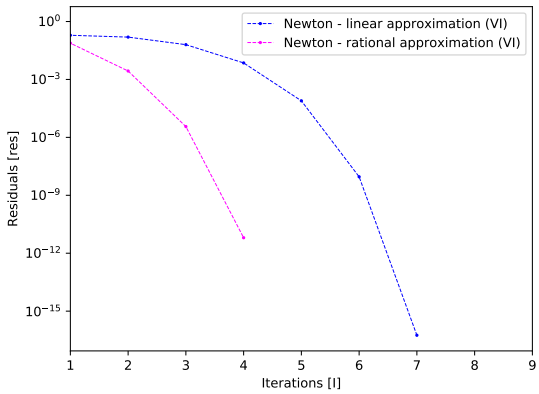
(c) Case 3



(d) Case 4

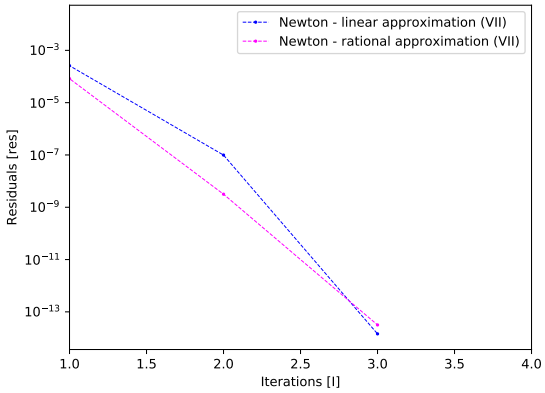


(e) Case 5

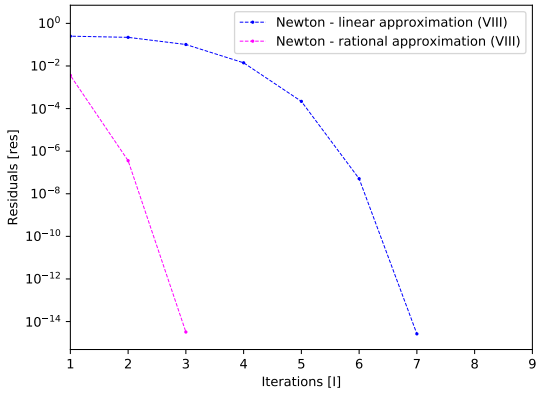


(f) Case 6

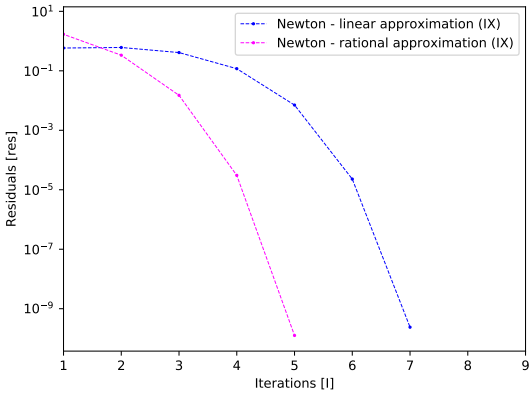
Figure 3: Comparison of linear and rational approximations for the manufactured functions.



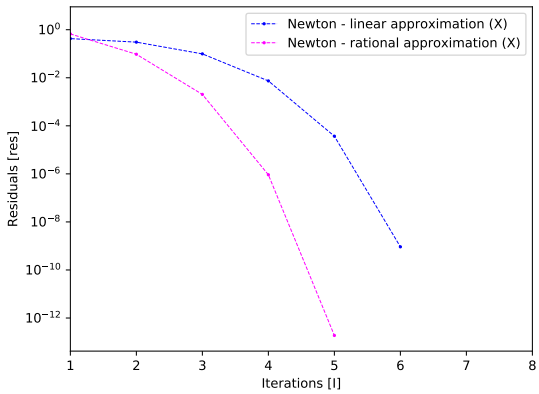
(a) Case 7



(b) Case 8



(c) Case 9



(d) Case 10

Figure 4: Comparison of linear and rational approximations for the manufactured functions.