Department of Electrical & Computer Engineering
1900 SW 4th Ave, suite 160
Portland, OR 97201

Portland State
UNIVERSITY

To: Midrar Adham, Professor

From: Jeff Dinsmore, Student

Subject: Newton Raphson Project

Date: Dec 3, 2025

---

1 Introduction

I was tasked with building a python script to run the Newton Raphson (N-R) power flow convergence to solve for different epsilon values. The program includes solving convergence using the Newton Raphson method as well as the Fast-Decoupled Load Flow (FDLF) method. Results were tabulated to compare the FDLF convergence and the N-R convergence using plots, tables and timing.

2 Methodology

I started with doing an outline of the message for the Newton Raphson power flow convergence. I then created a tiny amount of pseudo code. The next step was to write code based on the pseudocode steps, one function at a time.

After writing each function I would test the code to see if it worked. After all the functions were programmed, I ran a full code test to see if it converged with epsilon equals 0.01. I then changed epsilon to equal 0.00001 one to make sure it still converged.

Once everything looked good, then I moved forward to create an array of epsilon values from .01 to one 1e-9 and updated the iterations to include running through each epsilon value.

Plots were then added in along with tables and timing data for each iteration. If time allows, add in the FDLF method and compare to the N-R method.

2.1 N-R Algorithm Progress Update

2.1.1 Process Flow

1. Set up Y-bus matrix
2. Impose Flat Start Values
3. Set up mismatch matrix
4. Create Jacobian matrix
5. Solve for new δ and new voltage V
6. Check if ΔQ_i for the PV buses do not exceed the Q-limit
   (a) if ΔQ_i is outside the limit, convert PV bus to a PQ bus
7. Check $[V_i^{(k-1)} - V_i^{(k)} < \epsilon]$, $[P_i^{(k-1)} - P_i^{(k)} > \epsilon]$, $[Q_i^{(k-1)} - Q_i^{(k)} > \epsilon]$
   (a) If $V_i^{(k-1)} - V_i^{(k)} > \epsilon$ then use new δ and new voltage V in next iteration
   (b) If $V_i^{(k-1)} - V_i^{(k)} < \epsilon$ &
   $P_i^{(k-1)} - P_i^{(k)} > \epsilon$ &
   $Q_i^{(k-1)} - Q_i^{(k)} > \epsilon$, then N-R is solved and program can stop
8. Add Plots and tables in the final convergence

This table shows the workflow timeline for completing the N-R method.

	Anticipated Start Date	**Anticipated End Date**	**Actual Start Date**	**Actual End Date**
Process Flow Diagram	Nov 8 2025	Nov 8 2025	Nov 8 2025	Nov 9 2025
Pseudocode	Nov 9 2025	Nov 9 2025	Nov 9 2025	Nov 9 2025
Code	Nov 15 2025	Nov 16 2025	Nov 9 2025	Dec 3 2025
Func: Y-Bus	Nov 15 2025	Nov 15 2025	Nov 9 2025	Nov 12 2025
Func: Diag-El	Nov 16 2025	Nov 16 2025	Nov 9 2025	Nov 12 2025
Func: off-Diag	Nov 16 2025	Nov 16 2025	Nov 9 2025	Nov 12 2025
Test Procedures	Nov 23 2025	Nov 24 2025	Nov 28 2025	Dec 1 2025
Test Results	Nov 25 2025	Nov 25 2025	Nov 28 2025	Dec 2 2025
Algorithm Assessment	Nov 30 2025	Dec 1 2025	Nov 30 2025	Dec 2 2025

3 Results

Below are several figures to demonstrate the N-R powerflow convergence. The first set are plots. These plots take the maximum value of one of the PQ buses per iteration to compare to the ϵ value. The values it checks are the P, Q, V of the PQ buses against the tolerance level of ϵ. There is plots of the FDLF convergence to show how it compares to the N-R convergence. The FDLF takes more iterations to converge compared to N-R but is much faster per iteration. One can see that as tolerance ϵ = 1e-9, that the plots between the N-R and the FDLF look more and more similar.

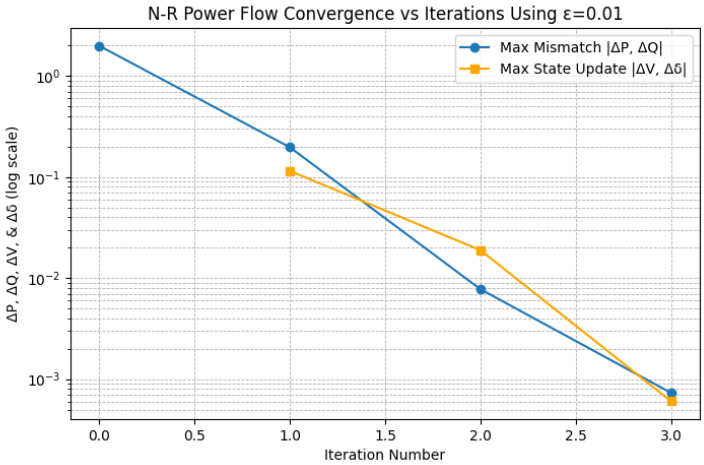

Figure 1: Newton Raphson (N-R) Power Flow convergence using ϵ = 0.01

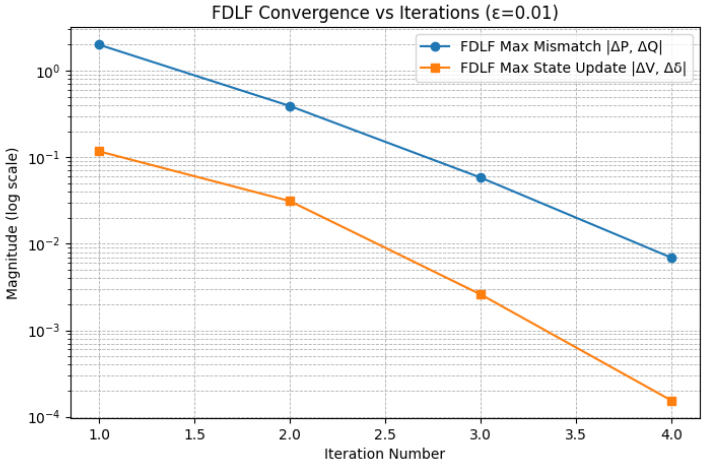

Figure 2: Fast-Decoupled Load Flow (FDLF) Power Flow convergence using ϵ = 0.01

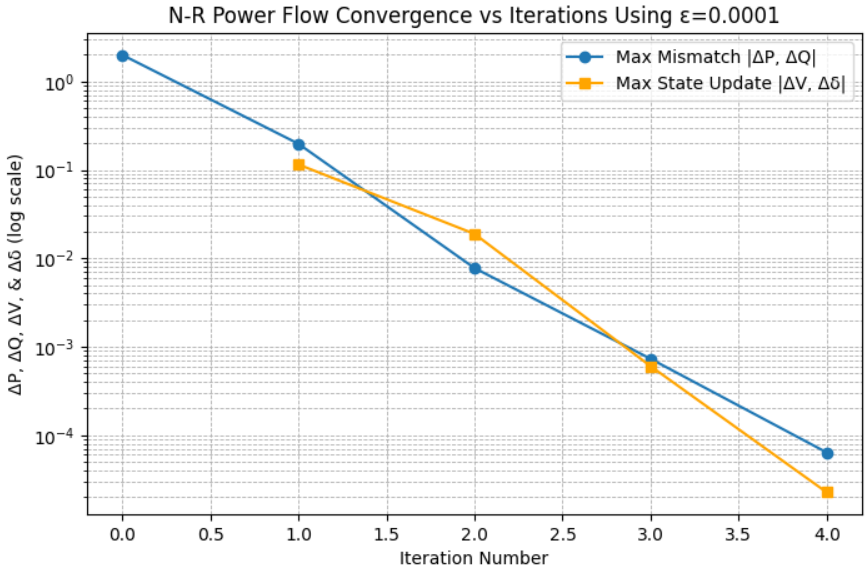

Figure 3: N-R Power Flow convergence using ϵ = 1e-4

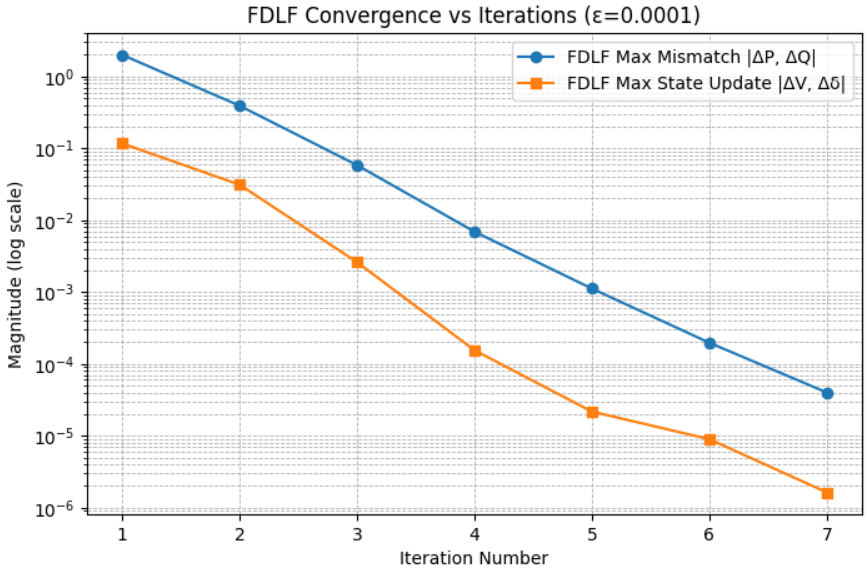

Figure 4: FDLF Power Flow convergence using ϵ = 1e-4

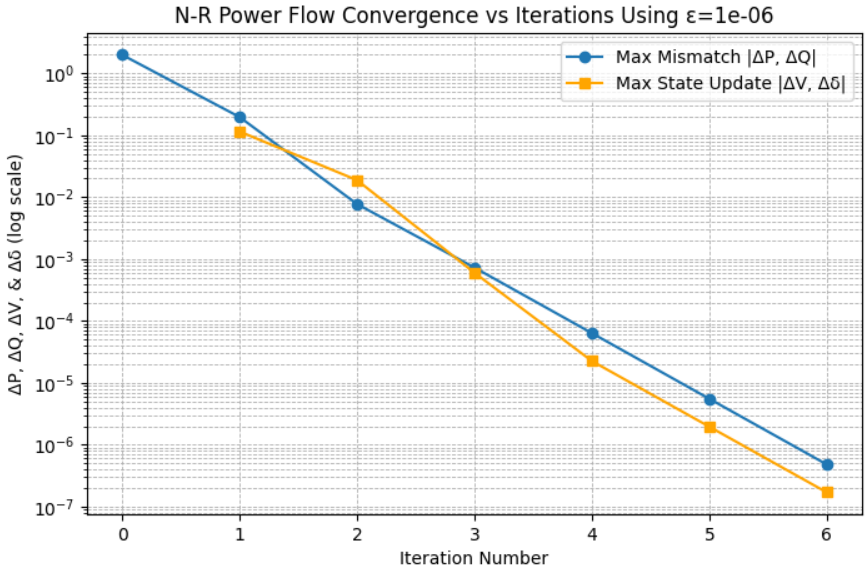

Figure 5: N-R Power Flow convergence using ϵ = 1e-6

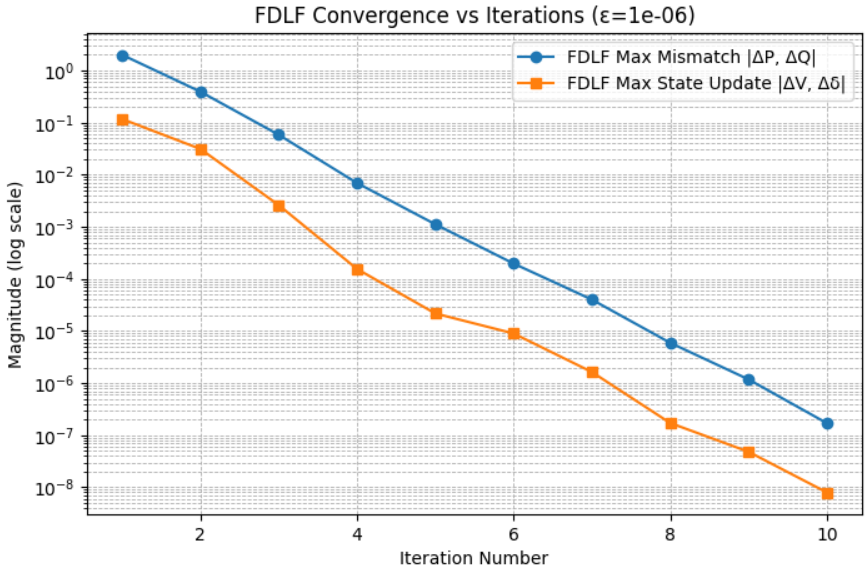

Figure 6: FDLF Power Flow convergence using ϵ = 1e-6

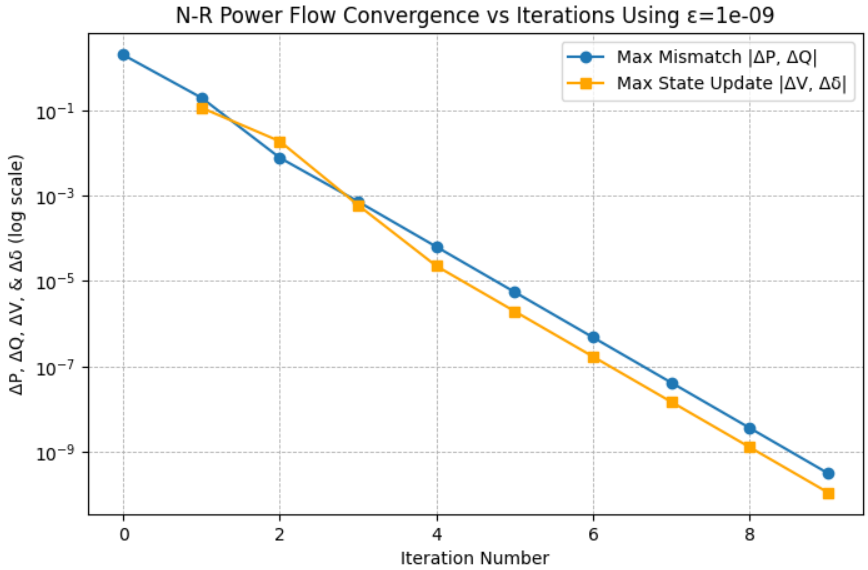

Figure 7: N-R Power Flow convergence using ϵ = 1e-9

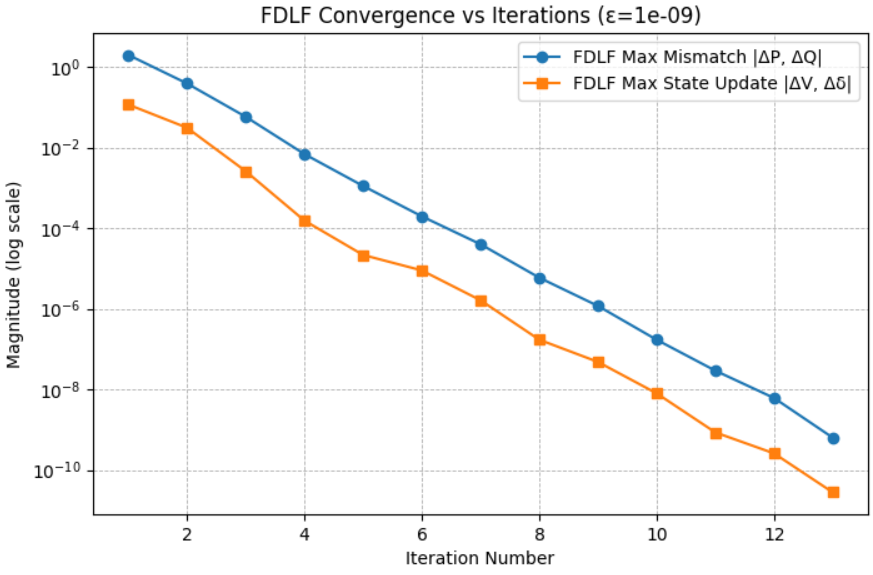

Figure 8: FDLF Power Flow convergence using ϵ = 1e-9

The following figures are the Voltage figures of the five buses. These figures show how the PQ buses went from 1.0pu to about 0.86 to 0.89pu upon convergence. It is expected to see the PV and Slack buses stay steady at 1.0pu and 0.98pu respectively.

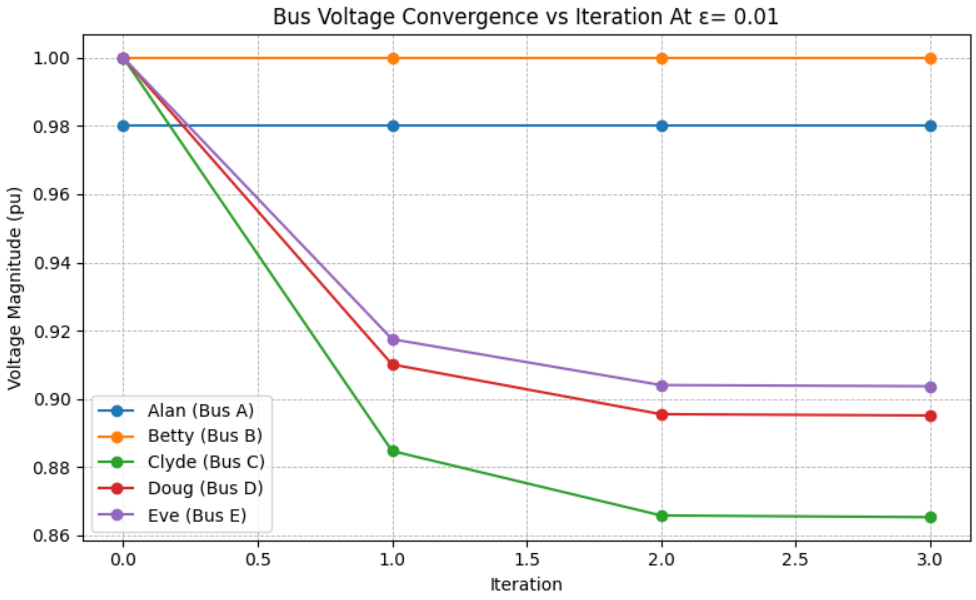

Figure 9: N-R voltage PQ bus convergence using ϵ = 1e-2

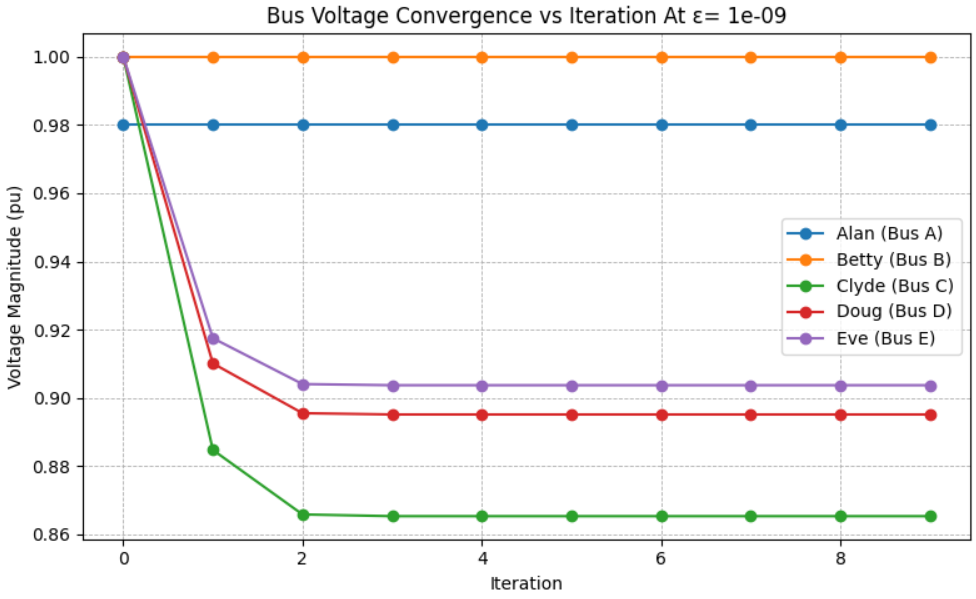

Figure 10: N-R voltage PQ bus convergence using ϵ = 1e-9

3.1 Convergence Tables

The tables below show how the N-R converges for each bus. The tolerance value of epsilon that was used for these tables was ϵ = 1e-6. For the SL slack bus and the PV bus, there is not changes in the voltage since they are not part of the convergence calculations. There is a change in the δ angle for the PV bus. One can see that the P_inj and Q_inj converge quickly in about three or four iterations using for PQ buses ϵ = 1e-6.

Bus	Name	Type	Iter	V (pu)	δ (deg)	P_inj (pu)	Q_inj (pu)	P_inj (MW)	Q_inj (MVAr)
1	Alan	SL	1	0.98	0	1.3799	0.6328	138	63.3
1	Alan	SL	2	0.98	0	1.5966	0.8709	159.7	87.1
1	Alan	SL	3	0.98	0	1.6071	0.876	160.7	87.6
1	Alan	SL	4	0.98	0	1.6078	0.8759	160.8	87.6
1	Alan	SL	5	0.98	0	1.6078	0.8759	160.8	87.6
1	Alan	SL	6	0.98	0	1.6078	0.8759	160.8	87.6

Figure 11: N-R Power Flow Slack Bus Alan, per iteration

Bus	Name	Type	Iter	V (pu)	δ (deg)	P_inj (pu)	Q_inj (pu)	P_inj (MW)	Q_inj (MVAr)
2	Betty	PV	1	1	-0.347132	1.9633	2.4585	196.3	245.9
2	Betty	PV	2	1	-0.494505	2.1039	2.814	210.4	281.4
2	Betty	PV	3	1	-0.50937	2.1007	2.8241	210.1	282.4
2	Betty	PV	4	1	-0.510662	2.1001	2.8243	210	282.4
2	Betty	PV	5	1	-0.510774	2.1	2.8243	210	282.4
2	Betty	PV	6	1	-0.510784	2.1	2.8243	210	282.4

Figure 12: N-R Power Flow PV Bus Betty, per iteration

Bus	Name	Type	Iter	V (pu)	δ (deg)	P_inj (pu)	Q_inj (pu)	P_inj (MW)	Q_inj (MVAr)
3	Clyde	PQ	1	0.884753	-5.73098	-0.9025	-0.7555	-90.3	-75.5
3	Clyde	PQ	2	0.865868	-6.67324	-1.0922	-0.8476	-109.2	-84.8
3	Clyde	PQ	3	0.865361	-6.70774	-1.1	-0.85	-110	-85
3	Clyde	PQ	4	0.865361	-6.70852	-1.1	-0.85	-110	-85
3	Clyde	PQ	5	0.865361	-6.70859	-1.1	-0.85	-110	-85
3	Clyde	PQ	6	0.865361	-6.70859	-1.1	-0.85	-110	-85

Figure 13: N-R Power Flow Slack Bus Clyde, per iteration

Bus	Name	Type	Iter	V (pu)	δ (deg)	P_inj (pu)	Q_inj (pu)	P_inj (MW)	Q_inj (MVAr)
4	Doug	PQ	1	0.910118	-4.24128	-0.9421	-0.7797	-94.2	-78
4	Doug	PQ	2	0.895534	-4.76468	-1.0007	-0.9458	-100.1	-94.6
4	Doug	PQ	3	0.89516	-4.7803	-1	-0.95	-100	-95
4	Doug	PQ	4	0.89516	-4.78087	-1	-0.95	-100	-95
4	Doug	PQ	5	0.89516	-4.78092	-1	-0.95	-100	-95
4	Doug	PQ	6	0.89516	-4.78092	-1	-0.95	-100	-95

Figure 14: N-R Power Flow PV Bus Doug, per iteration

Bus	Name	Type	Iter	V (pu)	δ (deg)	P_inj (pu)	Q_inj (pu)	P_inj (MW)	Q_inj (MVAr)
5	Eve	PQ	1	0.917507	-4.23409	-1.4174	-1.0301	-141.7	-103
5	Eve	PQ	2	0.904066	-4.77719	-1.5005	-1.196	-150	-119.6
5	Eve	PQ	3	0.903728	-4.79667	-1.5	-1.2	-150	-120
5	Eve	PQ	4	0.903728	-4.7976	-1.5	-1.2	-150	-120
5	Eve	PQ	5	0.903728	-4.79768	-1.5	-1.2	-150	-120
5	Eve	PQ	6	0.903728	-4.79768	-1.5	-1.2	-150	-120

Figure 15: N-R Power Flow PV Bus Eve, per iteration

3.2 Timing

Timing on the Newton Raphson versus the Fast-Decoupled Load Flow shows that the FDLF runs faster per iteration but slower overall because of the extra iterations needed to converge. The FDLF method was only marginally smaller in timing than the N-R method.

3.2.1 Time to process the convergence functions for N-R and FDLF

For 3 N-R iterations and 4 FDLF iterations at ϵ = 1e-02

- N-R iterations time: 0.382310 seconds taking 0.127437 seconds per iteration
- FDLF iteration time: 0.394830 seconds taking 0.098708 seconds per iteration

For 9 N-R iterations and 12 FDLF iterations at ϵ = 1e-09

- N-R iterations time: 0.144251 seconds taking 0.016028 seconds per iteration
- FDLF iteration time: 0.175566 seconds taking 0.013505 seconds per iteration

Total time to process four ϵ values, EPS = [1e-2, 1e-4, 1e-6, 1e-9]

- N-R total iterations time: 1.159153 seconds
- FDLF total iterations time: 1.179878 seconds

4 Conclusion

The results showed that although the FDLF ran faster per iteration because of less computations, the overall time to run to convergence was slower than the overall time for the N–R convergence. This was because more iterations were necessary for the FDLF then for the N-R.

Another surprising result was as the tolerance ϵ value became smaller and smaller the convergence patterns between the N-R and the FDLF started looking more similar to each other.

I was expecting the FDLF to be faster but the code is intertwined and some of the FDLF code is dependent on the N-R code. This may be the cause of the slower overall time despite the higher iterations of the FDLF.

While FDLF has its place perhaps with very large bus numbers to cut down on computations, for this project the way to go is using the N-R method.