295/395A Circuit Simulation and Optimization Methods: A Power Systems Perspective

Project 2: Positive Sequence Power Flow Analysis

Issued: February 28th, 2023 / March 6th 2023

Due Dates:

Checkpoint 0: March 9th, 2023, 11:59 PM EST

Checkpoint 1: March 21st , 2023, 11:59 PM EST

Final Submission: April 7th, 2023, 11:59 PM EST

OVERVIEW

In this project, you will develop a solver that conducts a positive sequence power flow analysis to solve standard transmission system networks (test cases).

Power flow analysis is a fixed frequency steady-state analysis that we use to solve for the states (e.g., complex bus voltages and currents) of the grid. Traditionally, we obtain the steady-state solution for the high-voltage transmission system via a positive sequence power flow analysis using the PQV formulation. Active power (P), reactive power (Q), and voltage are our system's state variables. In the PQV formulation, we solve two nonlinear power mismatch equations to obtain the system's bus voltage magnitudes and angles.

Rather than use the PQV formulation, you will create a power flow solver that uses the equivalent circuit formulation (ECF). In ECF, current (I) and voltage (V) are the state variables. We split power grid elements into their real and imaginary components and then solve the network as an equivalent circuit. During this project, you will see that one of the advantages of modeling the power grid as an equivalent circuit is incorporating circuit simulation methods to help obtain the steady-state solution of more extensive networks.

The development of your power flow analyzer consists of two parts. In Part 1, you will develop the solver, and in Part 2, you will add a limiting method to your solver to help find the solution of more extensive cases. The results for both portions of the assignment must be in your final project report. Report expectations are in *Deliverables*. The grading rubric for the project is in the *Grading* section. A recommended timeline to follow for completing Part 1 and Part 2 is in the *Suggested Timeline* section.

Course lectures and your assigned readings will help you complete this project. You are strongly encouraged to use these resources.

PART 1 – ASSEMBLING THE POWER FLOW SOLVER

Part 1 focuses on creating a complete power flow solver to solve small transmission networks. Your solver should include all elements of a power flow solver shown in Figure 1's flowchart.

Within your solver, you must do the following. First, build the positive sequence models. Then, systematically stamp the linear positive sequence models into the admittance matrix (Y). Finally, iteratively stamp the linearized circuits of the nonlinear elements into Y and solve the system using Newton-Raphson (NR).

In your project, you only need to implement the following positive sequence models in your code: generators, loads, shunts (fixed and switched), transmission lines (branches), and transformers. For the generators, you do not need to implement reactive power control.

As in project 1, you can use standard open-source Python libraries to develop your solution, such as NumPy and SciPy.

We have provided you with several python classes and functions to use in your implementation. These include a parser (*parsers/parser.py*) and empty models' classes (*models/*), most of which need to have functions completed for initialization, node/index assignment, and stamping. You also have a PowerFlow class (*scripts/PowerFlow.py*) where you will loop through the linear and nonlinear elements to stamp them, solve for your solution vector at iteration $k+1$ and check the error. The last function given is *Solve.py* (*scripts/Solve.py*) which handles initialization, postprocessing and controls the entire implementation. Within *Solve.py* and *PowerFlow.py,* there are specific step-by-step instructions that you should follow.

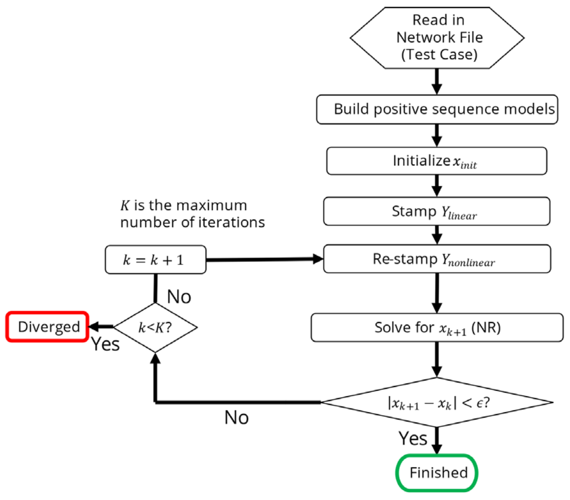

Figure 1. Basic structure of a positive sequence power-flow simulator.

You will run your solver by using *run_solver.py* and executing the *'python run_solver.py'* command in a terminal or command-line interface.

To test your solver, you will perform a power flow analysis on several common power systems test cases, including the GS-4, IEEE-14, IEEE-118, and ACTIVSg500. The test cases are in the RAW (.raw) and MATPOWER (.m) file formats in the Project 2 *testcases* folder. Table 1 has information about the power grid elements and initial conditions of the test cases. A description of each grid element's parameters is in the PSS/E Data Formats document (*docs/DataFormats.pdf*).

For each network, you will complete the power flow analysis using a prior solution as the initial conditions. Each network file has a *prior_solution* tag at the end of the file name (e.g., *testcases/IEEE-14_prior_solution.raw, testcases/IEEE-14_prior_solution.m*).

You will use MATPOWER's power flow tool to validate your implementation. MATPOWER is an open-source tool used to perform steady-state power systems analyses. Instructions for installing

MATPOWER and running a simple power flow are here. We encourage you to come to office hours if you have any difficulties with using MATPOWER.

In your project report, you should include the following:

1. Provide an overview of ECF power flow analysis in your own words and compare ECF to the traditional PQV formulation. You should include any formulas or equations derived for the project (including the derivations behind the stamps you used). In your report, you are encouraged to cite your assigned readings or lectures.
2. Give a brief explanation of your implementation and highlight your code's structure, classes, methods or functions, and anything unique about your codebase.
3. Computational performance of your solver using sparse vs. dense matrices. Does your solver's performance change?
4. For each test case:
    a. Report the maximum and minimum bus voltage magnitudes and angles.
    b. Record the average percent difference between your bus voltage magnitudes and angles and the MATPOWER solution.
    c. Report the number of Newton-Raphson iterations it takes for your implementation to complete a power flow analysis. Compare this to the number of iterations it takes for MATPOWER to solve the same case.
5. Any lessons learned implementing the positive sequence power flow solver.

Please **prioritize item 1 of the report above all other things**. We **will not** penalize you harshly for this assignment if you cannot get all your code to function correctly. We care more about developing a sound technical understanding of how power flow analysis should work and communicating your knowledge.

PART 2 – INTRODUCING CIRCUIT SIMULATION METHODS

For the second part of your project, you will implement voltage limiting. In voltage limiting, we limit the maximum allowable voltage step possible during an NR iteration.

In general, if our network has yet to converge, then before starting the next NR iteration, we execute our limiting method. Figure 2 illustrates where you should implement your limiting method within your power flow structure. A detailed explanation of voltage limiting is in section

IV of your reading Robust Power Flow and Three-Phase Power Flow Analyses (A. Pandey et al.) and lecture on limiting and homotopy.

To test your solver with voltage limiting, you will perform a power flow analysis on the PEGASE-9241and PEGASE-1359 using flat start initial conditions. Under flat start initial conditions, all bus voltage magnitudes are 1.0 p.u. All bus angles are zero. Each network file has a *flat_start* tag at the end of the file name (e.g., *testcases/PEGASE-9241_flat_start.raw, testcases/PEGASE-9241_flat_start.m*).

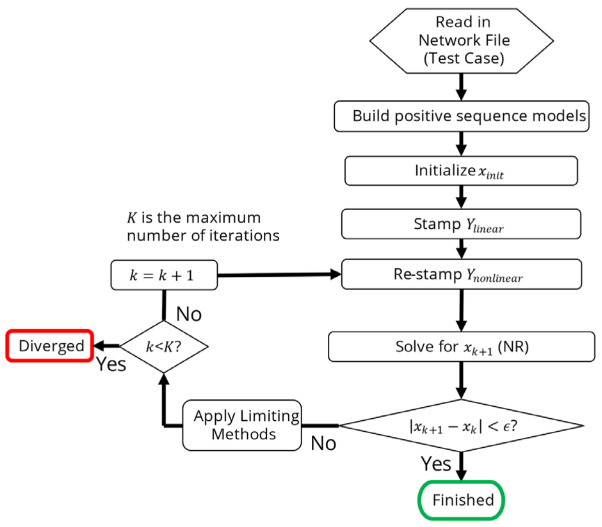

Figure 2. Positive sequence power flow simulator with limiting methods.

In your project report, you should include the following:

1. Describe voltage limiting in your own words and communicate the value of heuristics and limiting methods. Please also include a brief explanation of your implementation of voltage limiting. In your report, you are encouraged to cite your assigned readings or lectures.
2. Computational performance of your solver using sparse vs. dense matrices. Does your solver's performance change?
3. For each test case:
   a. Report the number of Newton-Raphson iterations it takes for your implementation to complete a power flow analysis with voltage limiting.
   b. Compare the number of iterations your implementation takes using voltage limiting to MATPOWER.
   c. Report the maximum and minimum bus voltage magnitudes and angles.
   d. Record the average percent difference between your bus voltage magnitudes and angles and the MATPOWER solution.
4. Any lessons you learned from implementing voltage limiting.

Please **prioritize item 1 of the report above all other things**. We will not penalize you harshly for this assignment if you cannot get all your code to function correctly. We care more about developing a sound technical understanding of how circuit simulation methods work and communicating that.

Table 1. Overview of the initial conditions and power grid elements in each of the Project 2 network cases.

Case Name	**Initial Conditions**	**# Buses**	**# Generators**	**# Loads**	**# Trans. Lines (Branches)**	**# Fixed Shunts**	**# Switched Shunts**	**# Transformers**
GS-4	Prior Solution	4	2	4	4	0	0	0
IEEE-14	Prior Solution	14	5	11	17	1	0	3
IEEE-118	Prior Solution	118	54	99	170	14	0	9
ACTIVSg500	Prior Solution	500	90	206	468	0	17	131
PEGASE-9241	Flat Start	9,241	1,445	4,895	13,797	0	7,326	2,252
PEGASE-13659	Flat Start	13,659	4,092	5,554	13,792	0	8,754	6,675

BONUS (10 PTS) – INTRODUCING A HOMOTOPY METHOD

The bonus task for Project 2 is to add homotopy to your power flow solver. In homotopy, we break our original hard-to-solve problem into a series of more manageable to solve sub-problems. We start by solving a trivial version of the problem and then gradually work our way back to the hard-to-solve problem using our solution of each sub-problem as the initial conditions for the next problem.

To implement a homotopy method, you will need to add additional stamps to some of your models and modify your code's framework. A detailed explanation of homotopy is in section IV of your reading Robust Power Flow and Three-Phase Power Flow Analyses (A. Pandey et al.) and Course Lecture.

The instructor and TA are available to provide you with more guidance on homotopy should you complete Part 1 and Part 2.

In your project report, you should include the following:

1. Describe homotopy in your own words and communicate the value of using homotopy methods. Please also include a brief explanation of your implementation of homotopy. In your report, you are encouraged to cite your assigned readings or lectures.
2. Computational performance of your solver using sparse vs. dense matrices. Does your solver's performance change?
3. For each test case:
    a. Report the number of Newton-Raphson iterations it takes for your implementation to complete a power flow analysis with homotopy.
4. Any lessons you learned from implementing homotopy.

Deliverables

Checkpoint 0 (March 9th, 2023)

In checkpoint 0, you will implement Newton-Raphson to solve the following set of nonlinear equations:

```
x^2 + y^2 - z^2 – 1 = 0
x^3 - y^3 + z^3 – 2 = 0
sin(x) + cos(y) + tan(z) – 3 = 0
```

When you solve the linearized equations in the loop, I want you to use sparse matrices:

- You can read about converting from dense to sparse matrices here: https://docs.scipy.org/doc/scipy/reference/generated/scipy.sparse.csr_matrix.html
    - Not the most efficient approach, but should be ok for now
- Next, instead of using dense matrix solver, which you have been using, you will have to use scipy function spsolve to solve for the linear equations iteratively: https://docs.scipy.org/doc/scipy/reference/generated/scipy.sparse.linalg.spsolve.html

To use Newton Raphson, you must calculate the Jacobian elements in each iteration and use the equations in slides 20 through 29 in Lecture 2.

More specifically, you will use the following equation:

$$F\left(x^{k+1}\right) \approx F\left(x^{k}\right) + F'\left(x^{k}\right) \cdot \left(x^{k+1} - x^{k}\right) = 0$$

over and over again until you find the solution.

Checkpoint 1 (March 24, 2023) – **Might be different for undergraduate students**

Submit code that can successfully run the GS-4 test case. In order for this case to run, you will need to have completed implementation for handling branches (non-transformer), loads, and generators. (See suggested timeline below for a roadmap of how to plan your implementation.)

Final Submission (April 7, 2023) - **Might be different for undergraduate students**

For the final submission of this project, you must provide two deliverables:

1. Submit your Python code for your simulator as a forked repository of the main course GitHub repository. You must push the final version of your code and submit a merge request by the deadline. More details about the submission process are in the GitHub repository. We will not grade any changes made after the deadline.
    a. You will notice that some of the code already has comments. We expect you to add your comments to the code to fill-in any missing details. Do not rely solely on the comments currently in the code.
2. Upload a project report in **.pdf* format to Canvas that:

a. Provides a brief explanation of your code’s structure, classes, methods, functions, and anything else that is unique about your codebase.
b. Includes your responses to all the tasks outlined in Part 1 and Part 2

Suggested Timeline

Start Date	End Date	Tasks
March 6	March 13	● Familiarize yourself with the elements of a '.RAW' file. ● Initialize all the empty model classes in the *models* directory in the code. ● Examine and ask questions about the starter code that is given to you in the *Project2* GitHub branch. ● Learn how to use MATPOWER and run a power flow simulation in MATPOWER.
March 13	March 20	● Complete the function to initialize your solution vector *v_init*. ● Add stamps for branches, loads, and generators to your code. ● Finish the *run_powerflow() and process_results()* functions. ● Successfully run the GS-4 test case.
March 20	March 27	● Add stamps for transformers and shunts to your code. ● Successfully run the IEEE-14, IEEE-118, and ACTIVSg500 test cases.
March 27	April 3	● Start the project report. ● Implement voltage limiting. ● Successfully run the PEGASE-9241and PEGASE-13659.
April 3	April 7	● Finalize project report and submit final code.

Grading

We will grade your project submission based on the rubric below.

Component	Sophisticated	Competent	Needs Work
	5 points	3 points	0 points
Checkpoint 1 Completion (5 points)	Submit code that can successfully run the GS-4 case and obtain a correct solution.	Code turned in where significant progress has been made (stamping functions completed for loads, generators, and branches).	Not turned in.
	26 – 30 pts	20 – 25 pts	0 – 19 pts
Solver Implementation and Accuracy (30 points)	The solution implements all power flow simulator elements correctly. The solver always works correctly, and the results are reasonable. Code has clear and sufficient comments.	The solution implements most power flow simulator components correctly. The solver works most of the time correctly, and the results are generally reasonable. The code is mostly logical and easy to follow.	Significant elements of the power flow simulator are missing. The program does not often work correctly or work at all. No results or unrealistic results. Few to no comments make the code hard to understand.
	31 – 35 pts	26 – 30 pts	0 – 26 pts
Power Flow Comprehension and Understanding (35 points)	The student effectively communicates what power flow analysis is, how to perform a power flow analysis and the advantage of solving power flow using the Equivalent Circuit Formulation.	The student can communicate what power flow analysis is and the advantage of solving power flow using the Equivalent Circuit Formulation. Still, they neither fully comprehend how it works or the steps to perform a power flow analysis.	The student cannot communicate what power flow analysis is or demonstrates little understanding of performing a power flow analysis using the Equivalent Circuit Formulation.
	16 – 20 pts	10 – 15 pts	0 – 9 pts
Circuit Simulation Method Comprehension and Understanding (20 points)	The student demonstrates an understanding of what circuit simulation methods are, how to implement voltage limiting, and the impact circuit simulation methods can have on analysis.	The student demonstrates an understanding of what circuit simulation methods are and their impact but does not fully comprehend how to implement voltage limiting.	The student demonstrates little to no understanding of circuit simulation methods, voltage limiting, or the impact of circuit simulation methods.
	8 – 10 pts	5 – 7 pts	0 – 4 pts
Written Communication (10 points)	The report is well-organized and coherent. All figures and tables are readable, appropriately sized, and properly labeled, and captioned correctly. There are no spelling or grammar errors.	The report is, for the most part, well-organized and coherent. Some figures and tables are hard to follow or labeled and captioned inappropriately. There are a few spelling and grammatical errors, but they do not detract from the work.	The report has little to no structure or organization. No labels and captions on figures and tables. Many spelling and grammatical errors make the report difficult to read.