PowerEdu.jl - A Julia Package for Teaching Power System Courses

Ninad Gaikwad, Aryan Jha, Sajjad Uddin Mahmud and Anamika Dubey
School of Electrical Engineering & Computer Science, Washington State University, Pullman, WA, USA
{ninad.gaikwad, aryan.jha, sajjaduddin.mahmud, anamika.dubey}@wsu.edu

***Abstract*—We introduce PowerEdu.jl, an open-source, beginner-friendly package in the Julia programming language designed for budding power system engineers. This package addresses the current gap in accessible and comprehensive tools for transmission network computations. PowerEdu.jl covers Power Flow using innovative dense and sparse data structures, Continuation Power Flow, State Estimation, Optimal Power Flow, Small-Signal Stability, and Transient Stability Analysis. Notably, the package is scalable, allowing for analysis of systems of varying sizes. User interaction with component modules is highly customizable; for example, users can opt to print detailed intermediary steps, such as Jacobians and mismatches, in Power Flow calculations. We use DataFrames for intuitive and visually appealing data representation. In this paper, we detail the key modules of PowerEdu.jl, elaborate on the special data structures implemented, and demonstrate the breadth and flexibility of algorithm customization available to users. Our package has been rigorously validated against established benchmarks, affirming its reliability and effectiveness as a powerful training tool for the next generation of power system engineers. Mention the network and key finding in one line.**

Why Julia?

***Index Terms*—Julia, Open-Source Tools, Power System Analysis, Power System Education**

I. Introduction

Power System Analysis of transmission systems requires competencies in a variety of aspects. Power System Analysis, which has traditionally associated with quasi-steady state studies, encompasses aspects such as Power Flow, Continuation Power Flow, Economic Dispatch, Optimal Power Flow, State Estimation and so on. Each of these aspects requires a broad knowledge of Mathematics and Physics but also a wide variety of skillsets, including strong programming skills, in order to actually test novel algorithms and bringing essential control schemes to fruition. There can be a significant gap between understanding of the theory and actual implementation for Power System studies and day-to-day operation. Aspects like Sparse Power Flow which requires usage of special data structures of the same name especially highlight how actual implementation can vary from textbook algorithms, which are often written in pseudo code. While academic curricula and textbooks provide a strong foundation in theory, there remains a pressing need for practical tools that translate these theoretical underpinnings into tangible, implementable solutions. This gap becomes particularly evident when students or new professionals are tasked with the direct application of these theories in real-world scenarios. While various open-source packages exist to aid power system researchers, many are designed with a primary focus on delivering end results. These tools may not be as accommodating for newcomers who are eager to understand the underlying processes, delve deep into the intricacies of algorithms, or get their hands dirty with the code. Our software acknowledges this gap. Recognizing the educational journey many newcomers embark upon, we've crafted our package to not only deliver accurate results but also facilitate a deeper understanding. Users can easily access internal variables during iterative processes, something many other packages shield away. Furthermore, PowerEdu.jl stands out for its high level of customizability, allowing individuals to tinker, modify, and adapt algorithms to their specific needs. For those passionate about 'coding it up' and truly comprehending the nuts and bolts of power system analyses, our package offers a unique, hands-on experience. Our free and open-source package, PowerEdu.jl aims to serve as a bridge for budding power system engineers who may find the initial stages of coding and computational analysis challenging. By offering an accessible, well-documented and easy to tinker platform, we aim to narrow the gap between newcomers to the field and seasoned experts who have dedicated years at renowned national laboratories or corporations, developing sophisticated software tools utilized by the industry.

II. Theory and Scenarios

A. Power Flow

B. Sparse Power Flow

C. Continuation Power Flow

D. State Estimation

E. Optimal Power Flow

III. Description of Modules

A. Power Flow

B. Sparse Power Flow

For Transmission Networks, most of the commonly used data structures for analyses are sparse in nature, i.e. most of their elements are zero. Data Structures such as Y_{Bus}, Jacobian J, the LU Factors of the Jacobian LU are sparse in nature. The sparsity only increases as the size of the system increases. ¡Insert some values of sparsity for different transmission systems¿. This sparsity can be exploited for faster computation and smaller data storage requirements, when

performing analysis w.r.t. any aspect of Power Systems. For example, using taking advantage of the sparsity of the above mentioned data structures, along with other schemes such as parallel computation and Single Instruction Multiple Data (SIMD) operations, the authors of [1] were able to perform very fast Newton Raphson Power Flow for large transmission systems.

C. Continuation Power Flow

D. State Estimation

E. Optimal Power Flow

F. CDF Parser

IV. User Interface

Upon donwloading PowerEdu.jl on their machine, user will interact with the following directory heirarchy. For the sake of clarity, folders pertaining only to the IEEE_14 Bus test case are shown, however, in general, every test case will have its dedicated folders for inputs and outputs.

A. Directory Structure

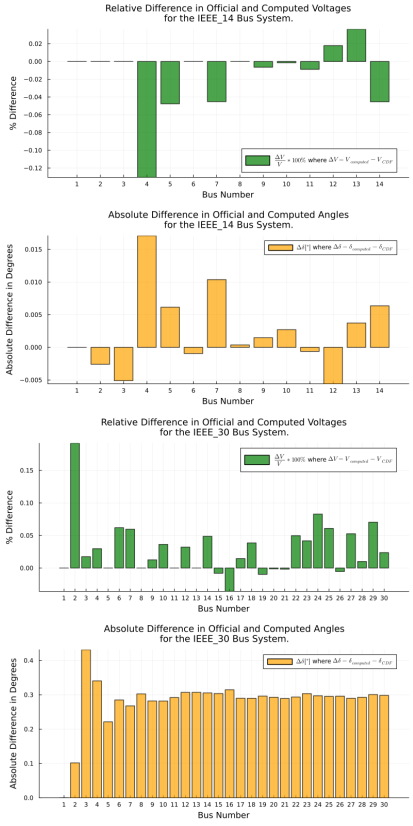
```
root (PowerEdu)
├── data
│   └── IEEE_14
│       └── IEEE_14_Data.txt
├── processedData
│   └── IEEE_14
│       ├── BusDataCard_pu.csv
│       ├── BranchDataCard_pu.csv
│       ├── YBus.csv
│       └── ... (other generated files)
├── src
│   ├── ContinuationPowerFlow.jl
│   ├── IEEE_CDF_Parser.jl
│   ├── OptimalPowerFlow.jl
│   ├── PowerFlow.jl
│   ├── StateEstimation.jl
│   ├── SparsePowerFlow.jl
│   └── ... (other modules)
├── main.jl
├── README.md
└── LICENSE
```

B. Pluto Notebook Environment

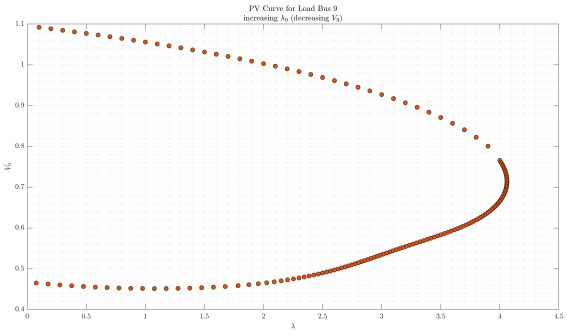
V. Results

A. Power Flow

B. Sparse Power Flow



C. Continuation Power Flow



D. State Estimation

E. Optimal Power Flow

VI. Conclusion

[2], [3], [4], [5]

References

[1] A. Ahmadi, M. C. Smith, E. R. Collins, V. Dargahi, and S. Jin, "Fast Newton-Raphson Power Flow Analysis Based on Sparse Techniques and Parallel Processing," *IEEE Trans. Power Syst.*, vol. 37, no. 3, pp. 1695–1705, Sep. 2021.

[2] M. L. Crow, *Computational Methods for Electric Power Systems (Electric Power Engineering Series)*. Boca Raton, FL, USA: CRC Press, Jun. 2021. [Online]. Available: https://www.amazon.com/Computational-Methods-Electric-Systems-Engineering/dp/1032098228
[3] "Nrel-sienna github.com," accessed: 2023-05-29. [Online]. Available: https://github.com/NREL-Sienna
[4] J. J. Grainger and W. D. Stevenson Jr, *Power system analysis*. McGraw-Hill series in electrical and computer engineering, 1994.
[5] "Nrel sienna," accessed: 2023-04-21. [Online]. Available: https://www.eia.gov/consumption/