Fast Batched Solution for Real-Time Optimal Power Flow With Penetration of Renewable Energy

James Peavy, Mert Akyurekli and Vishal Sivaraman

ARTICLE INFO

Keywords:
CUDA
OPF
PDIPM

ABSTRACT

Real Time Optimal Power Flow problems are increasing in relevance because of the rise in the use of renewable energy generators in power systems. Because of uncertainties in the renewable energy sources, with solar and wind being the focus of the paper, it becomes important to re-solve the power flow problem using the most recent data to predict the future in shorter horizons. This paper proposes a novel way to find a hot-start to the problem by running the OPF problem for multiple possible future scenarios so that we get close to the actual reality in one of them when the uncertainty is realised. The paper describes a parallelised Primal Dual Interior Point Method as a solution method for the problem. The parallelised method ensures the runs complete in a reasonable timescale as is required by RTOPF problems. We did an extensive literature review, attempted the CPU version of the algorithm and outlined our plans on how we are going to proceed further.

1. Introduction and Literature Review

The Optimal Power Flow problem was introduced in Carpentier (1962). It is defined as "the determination of the complete state of a power system corresponding to the best operation within security constraints" in Carpentier (1979). In simpler terms, it ensures that the electricity generation companies meet the forecasted demand with the least cost and at the same time ensure generation and transmission safety constraints are taken care of. For example (as we will see later in the problem formulation), there will be constraints on how much you can ramp up production and on how much you can generate to avoid overloading the power lines. These have been traditionally solved with derivative-based methods (including KKT conditions) as described in the review (Carpentier (1979)).

Real-Time Optimal Power Flow (RTOPF) problem was introduced in Tang, Dvijotham and Low (2017). It was established in the context to handle multiple energy resources where we need to handle lot of uncertainties - especially in the presence of renewable energy generators. The essential difference between optimal power flow and real time optimal powerflow is the fact that we are focused on ensuring the operating constraints are satisfied but we are willing to concede on the objective (which is the power generation cost) in order to ensure we quickly solve the problem. The work used quasi-Newton methods to compute sub-optimal updates to the solution so that it can operate under shorter timescales. Specifically it used the Limited memory BFGS algorithm developed by Liu and Nocedal (1989). The implementation was in *MATLAB*. This method of solving has been termed as "OPF-based energy management" in Mohagheghi, Alramlawi, Gabash and Li (2018). The article also talks about a classification of methods called "constraint satisfaction-based real-time energy management" and highlights the salient features of these two broad classes. As the name suggests the constraints satisfaction class focuses entirely on finding a basic feasible solution for the given constraints at hand. With the renewable energy generators being an important integral part of modern grid networks as the we move towards using greener sources of energy it becomes crucial to address this problem to ensure optimal operations (Huang and Dinavahi (2018a)). The natural questions that arise in one's mind now is,

- can we make the solution better (more optimal - in terms of the costs or in terms of set-point tracking)?
- can we compute the solution faster?

Researchers have attacked both these questions (often only one at a time). Woo, wu, Park and Roh (2020) use a Deep Reinforcement Learning method called TD3 which itself is an extension of Deep Deterministic Policy Gradient (DDPG) (Silver, Lever, Heess, Degris, Wierstra and Riedmiller (2014)). Here, they use a Gaussian noise on the loads to model uncertainties of the renewable energy generators. A safe Deep Reinforcement Learning method has been proposed in Wu, Chen, Lai and Zhong (2023) where by delinking the constraint violation penalty and the economic cost reward, they remediate the reward sparsity issue. Once the model is trained, the calculations for power flow are two orders of magnitude faster compared to solving the RT-OPF optimization problem. Solving of optimization problems can be accelerated by using Graphical Processing Units (GPUs). The amount of speed-up that can be done is constrained by by Amdahl's law (Amdahl (1967)):

$$S = \frac{1}{F_S + \frac{F_P}{P}} \tag{1}$$

where,

- S is the Speed-up
- P is the number of processors
- F_S is the fraction of serial work
- F_P is the fraction of parallel work

Interior-point methods are a class of solution methods used in solving the RT-OPF. They use barrier functions in order to ensure the feasibility of the inequality constraints (Mohagheghi et al. (2018)). Oliveira, Oliveira, Pereira, Honório, Silva and Marcato (2015) use a safety barrier parameter δ to enhance the convergence of such methods, and even if the problem is infeasible, they provide directions for operators to bring it back to feasible space. Predictor-corrector interior-point method is used by Rakai and Rosehart (2014) for solving the RTOPF problem where the most computationally expensive step of the algorithm - matrix factorization is parallelised. They use the parallel computing toolbox of MATLAB, so there will be some run time overheads since they did not use a low level language like C to program the logic. Su, He, Liu and Wu (2020) introduce a parallelised implementation of Newton-Raphson method for solving the problem. They analyze the sparsity of the problem and identify areas where specific algorithms which work well for sparsity and GPU vectorization can be applied. For larger systems ($>$ 10000 buses), Li, Li, Yuan, Cui and Hu (2017) achieved $\approx 3x$ speedups using GPUs. It integrates the inexact Newton methods and the preconditioned iterative conjugate gradient method to solve the OPF problem. The preconditioning logic is ported to GPUs. A meta-heuristics based method using Particle Swarm Optimization is proposed in Roberge, Tarbouchi and Okou (2016). Given that it is a highly parallelizable algorithm and the author's exploited the sparseness of the problem well to provide $\approx 17x$ speedups compared to CPU. Wang, Wende-von Berg and Braun (2021) presents a nice heterogeneous algorithm which uses both CPU and GPU for Newton-Raphson (specifically in LU factorization/re-factorization steps of the matrices).

The paper we are implementing - "Fast Batched Solution for Real-Time Optimal Power Flow With Penetration of Renewable Energy" (Huang and Dinavahi (2018a)) presents a two-pronged improvement of both reducing wait time and increasing accuracy of the solution which is not usually the case. This is achieved by parallelizing the solution procedure which is an interior-point method. The uncertainty of renewable energy is dealt with by generating multiple plausible future scenarios. Although this prediction aspect is not relevant to the course, we want to make a couple of brief mentions. Mohagheghi, Gabash and Li (2017) develop the method this work uses for generating scenarios for wind energy. The error is assumed to be Beta distributed and scenarios are sampled from that. The distribution is calibrated using historical data. However, they use the BONMIN solver in GAMS to solve the optimization problem. Solar energy is modelled by a bi-modal distribution in Reddy (2017) and they use Genetic Algorithm to solve the OPF problem. We solve each of these scenarios and maintain a look-up table. Once the actual scenario happens, we quickly resolve the problem by using the closest available scenario as a hot-start. Since all this is happening in real-time, the challenge is to solve multiple scenarios as quickly as possible. The exact procedure is described in the methods section.

We also found some more recent work that cite this work, improve some aspects of it - either extending the application to more general renewable energy systems or adapt the parallel implementation to specific systems. Mohagheghi, Alramlawi, Gabash, Blaabjerg and Li (2020) follows a similar scenario generation and maintains a look-up table for hot-starting the actual problem but sets up a Mixed Integer Non-Linear Program (MINLP) instead of a quadratic program. Example of integer variables include number of charge-discharge cycle of the battery. GAMS solvers are used to solve the problems. The authors of this work have also developed GPU accelerated Newton-Raphson methods in Huang and Dinavahi (2018b). There has also been a multi-objective problem formulated with a solution strategy in Chen, Qian, Zhang and Sun (2019) where other aspects of power generation like emissions are included as a part of the objective. Some other recent work which could be of interest include Shin, Pacaud and Anitescu (2024) which introduces a non-linear program with a completely parallelisable solution framework achieving 5x speedups. They are able to do this by porting sparse automatic differentiation (AD) and sparse linear solver routines to GPUs.

This interim report has been divided into four sections. The second section elaborates on the problem formulation and solution strategy along with description of our code. The third section discusses the results we obtained and how we stand with respect to the original work. The final section presents our conclusions and our strategy for the rest of the project.

2. Methods

The following is the optimization problem set up for the RTOPF.

Objective Function:

$$\min F = \sum\nolimits_{g\in\mathcal{N}_g} \left[a_g \left(P_g^G \right)^2 + b_g P_g^G + c_g \right]. \tag{2}$$

Nodal Power Balance Constraints

$$P_i^G - P_i^D = V_i \sum_{j\in\mathcal{N}_b} V_j (G_{ij}\cos\theta_{ij} + B_{ij}\sin\theta_{ij}), \tag{3}$$

$$Q_i^G - Q_i^D = V_i \sum_{j\in\mathcal{N}_b} V_j (G_{ij}\sin\theta_{ij} - B_{ij}\cos\theta_{ij}). \tag{4}$$

Generator Capacity Constraints

$$P_g^{G,min} \le P_g^G \le P_g^{G,max}, \tag{5}$$

$$Q_g^{G,min} \le Q_g^G \le Q_g^{G,max}. \tag{6}$$

Ramp Rate Constraints

$$P_g^{G0} - R_g^{G,down} \le P_g^G \le P_g^{G0} + R_g^{G,up}. \tag{7}$$

Voltage Deviation Constraints

$$V_i^{min} \leq V_i \leq V_i^{max}. \tag{8}$$

Line Security Constraints

$$f_{ij}^{min} \leq f_{ij} \leq f_{ij}^{max}. \tag{9}$$

which is effectively:

$$\theta_{ij}^{min} \leq \left(\theta_{ij} = \theta_i - \theta_j\right) \leq \theta_{ij}^{max}. \tag{10}$$

where,

- $\mathcal{N}_b, \mathcal{N}_l$ Set of buses and lines
- $\mathcal{N}_g, \mathcal{N}_r$ Set of buses where thermal and renewable generators are integrated.
- a_g, b_g, c_g Coefficients of the quadratic cost function of generator g.
- P_g^G, Q_g^G Active and reactive power output of thermal generator g.
- P_r^R, Q_r^R Active and reactive power output of REG r.
- P_i^D, Q_i^D Active and reactive power demand of bus i.
- G_{ij}, B_{ij} Transfer conductance and susceptance between buses i and j.
- V_i, V_j Voltages magnitude at node bus i and j.
- P_g^{G0} Active power output of generator g in the previous sub-interval.
- f_{ij} Power flow on line ij.
- θ_{ij} Voltage angle difference between buses i and j.
- θ_i, θ_j Voltage angle at node bus i and j.
- $^{down}, ^{up}$ Ramp down and up limits of thermal generator.
- $^{min}, ^{max}$ Lower and upper limits of specified variables

The entire formulation has been borrowed from the paper we are working on (Huang and Dinavahi (2018a)).
The decision variables here are P^G, Q^G, V_i and θ_i. It is important to note that some of these values must be computed using data not shown in this paper, specifically the G_{ij} and B_{ij} values. B_{ij} is given by the data, and G_{ij} is calculated using the following formula, where R_{ij} is the line resistance, X_{ij} is the line reactance, and $j = \sqrt{-1}$:

$$G_{ij} = \frac{1}{R_{ij} + jX_{ij}} - jB_{ij} \tag{11}$$

The above optimization problem is solved for multiple scenarios generated using the currently available renewable energy data. At a given state, the paper forecasts 1024 possible futures for generation of renewable energy: P_i^D. There are

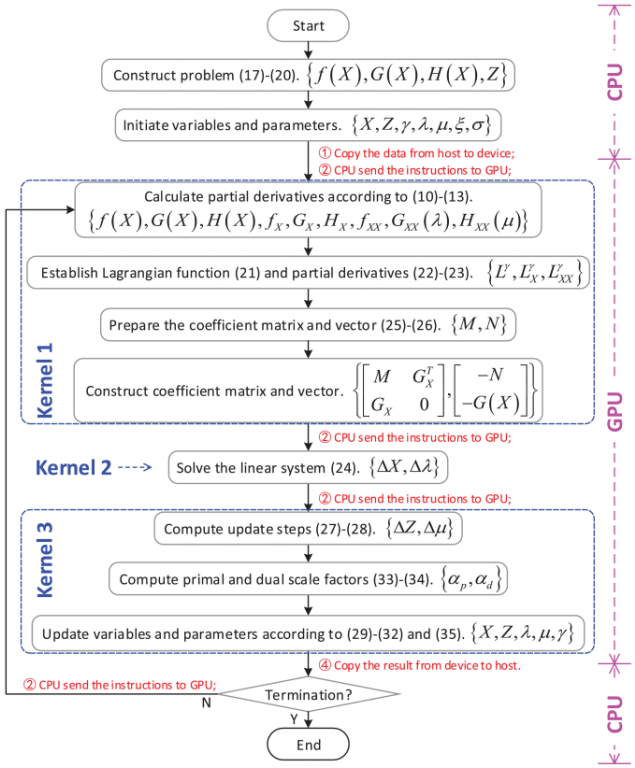

Figure 1: Kernel Setup

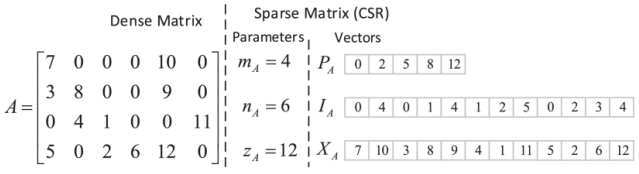

Figure 2: Showcase of CSR format. P_A being the Row Indexes, I_A being the Column Indexes, and X_A being the non-zero data, all in the Classic CSR format

therefore 1024 different RTOPF problems to solve. This is done in a parallel fashion, as seen in figure 1. The algorithm used is the Primal-Dual Interior Point Method (PDIPM) as mentioend in Wang, Murillo-Sanchez, Zimmerman and Thomas (2007). The inequality constraints are converted to equality constraints by bringing in the slack vector Z.

- **Kernel-1**: Calculation of partial derivatives of the objective and constraints. The derivatives are analytically computed and implemented in the code to do these computations. Further, the Lagrangians are computed and the matrix is generated. Additionally, as our matrices are stored in CSR format, matrix concatenation is a more complicated business. Pseudo-code in the original paper is to be translated into parallel code in kernel 1 as a part of the data preparation.
- **Kernel-2**: This is the most compute intensive set as it involves matrix inversion (solving of a linear system of equations). This is accomplished using the CuSolver.

- **Kernel-3**: The solved linear equation is to compute and evaluate the update steps for the decision variables for the next iteration

This process is repeated till the following occur:

- The (scaled) constraints are followed with a tolerance of ϵ
- The (scaled) Lagrangian function is close enough to zero with tolerance of ϵ
- The scaled change in objective is within ϵ

For this paper, we have tasked ourselves with the understanding of the basic problem and its implementation using cuSolver, therefore we will assume no REG data is yet added. Given the context of the course we believe the scenario generation itself is of no particular interest to us, therefore basic methods of scenario generation will be used and further discussed in the Conclusion. Our focus is going to be on implementing the optimization algorithm in parallel and comparing the GPU speed-ups. We also plan to solve the problem with the CPU using Csparse.

We have solved the basic problem in cvxpy in order to get an example on both the proper form of the solution, which is not discussed in the original paper, and the time increase we would expect for each version of the basic problem (14, 57, 118, and 300 Bus cases). We also used the MATPOWER package in MATLAB developed by Zimmerman, Murillo-Sánchez and Thomas (2011a). Specifically we used two interior point methods present in the package - MIPS (MATPOWER Interior Point Solver - Wang et al. (2007)) and PDIPM (Primal Dual Interior Point Method - Zimmerman, Murillo-Sánchez and Thomas (2011b)). It is important to note that PDIPM is the algorithm is used in the paper. These results are shown in the next section.

3. Results and Discussions

The performance metrics presented for the OSQP and SCS solvers in table 1 reflect efficiency in a singular scenario per case. In contrast, the original study conducted evaluations over 1024 scenarios to provide a comprehensive analysis. OSQP leverages operator splitting techniques to efficiently solve quadratic programs, demonstrating robustness and scalability in our tests. Python and MATLAB code is not one to one comparable because we solved a simpler problem in Python (initially) for our understanding whereas the MATLAB packages deal with the full fledged problem.

Results using MIPS and PDIPM are present in table 2. Firstly, we observe that the optimal objective value attained is same in both cases, which is good. Secondly, we see PDIPM, the algorithm used in the paper, beats the other popularly used solver MIPS by an order of magnitude in some cases. This is the expected line as commented on the MATPOWER user manual. PDIPM becomes better as we increase the bus sizes and hence the complexity of the system under consideration. The MATPOWER package code provides us a good starting point for us to get started with some of the aspects of the C implementation, especially for implementing the constraints and their derivatives.

Table 1
Performance comparison of optimization algorithms SCS and OSQP

Case	Compile Time (s)		Solver Time (s)	
	SCS	OSQP	SCS	OSQP
14	0.268	0.443	0.017	0.0089
57	3.43	3.5	0.035	0.012
118	8.37	7.48	0.15	0.0322
300	77.3	67.57	0.43	0.22

4. Conclusion and Path forward

Given the full problem understanding, we have a few tasks before us:

- **Create Kernels in C**
- **Implement REG data**
- **Create Scenarios using REG data**

For the Kernel creation, we will have to implement the data in a form C can read and can be solved. The original paper outlines a methodology for preparing the data in kernel 1 and 3, which will be the area of primary effort going forward.

For the implementation of REG data, we are looking to use solar information for a random day/time combination. The original paper outlines the fact that in very small timescales, even highly variable weather pattern data can have its variability drastically reduced. Using that fact, it doesn't truly matter which day or time. that being highly variable or highly regular data, we choose to analyse. We can then follow the original paper's plan of replacing some of the original IEEE thermal generators with 2, 4, 10, and 25 REGs for the 14, 57, 118, 300 bus cases respectively.

For the Scenario generation, we will likely choose to use a basic linear regression model to fit for the prediction of the next time-step, we can then pick random numbers between any given Prediction Interval to generate a given scenario. This will give us a number of scenarios to analyze. Logically, the number of REGs increases the amount of scenarios we must analyze exponentially with the following relation, $n_s^{n_r}$ where n_s is our number of scenarios per REG and n_r is the number of REGs we are implementing.

5. Data Sources

1. MATPOWER
   (a) Cases 14, 57, 118, 300

Table 2
Performance comparison of optimization algorithms MIPS and PDIPM

Case	Compute Time (s)		Optimal Objective ($/hr)	
	MIPS	PDIPM	MIPS	PDIPM
14	0.09	0.02	8081.53	8081.53
57	0.12	0.03	41 737.79	41 737.79
118	0.17	0.06	129 660.70	129 660.70
300	0.27	0.16	719 725.10	719 725.10

6. Case 14 Data

For illustrative purposes, we added the IEEE 14-bus test case values into the variables in our optimization expressions. All values are vectors, read from top to bottom, left to right.

$$P_g^{G0} = \begin{bmatrix} 232.4 & 40 & 0 & 0 & 0 \end{bmatrix}$$
$$Q_g^{G0} = \begin{bmatrix} -16.9 & 42.4 & 23.4 & 12.2 & 17.4 \end{bmatrix}$$
$$a_g = \begin{bmatrix} 0.0430292599 & 0.25 & 0.01 & 0.01 & 0.01 \end{bmatrix}$$
$$b_g = \begin{bmatrix} 20 & 20 & 40 & 40 & 40 \end{bmatrix}$$
$$c_g = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$
$$P_i^D = \begin{bmatrix} 0 & 21.7 & 94.2 & 47.8 & 7.6 & 11.2 & 0 \\ 0 & 29.5 & 9 & 3.5 & 6.1 & 13.5 & 14.9 \end{bmatrix}$$
$$Q_i^D = \begin{bmatrix} 0 & 12.7 & 19 & -3.9 & 1.6 & 7.5 & 0 \\ 0 & 16.6 & 5.8 & 1.8 & 1.6 & 5.8 & 5 \end{bmatrix}$$
$$V_i = \begin{bmatrix} 1.06 & 1.045 & 1.01 & 1.019 & 1.02 & 1.07 & 1.062 \\ 1.09 & 1.056 & 1.051 & 1.057 & 1.055 & 1.05 & 1.036 \end{bmatrix}$$
$$V_i^{min} = \begin{bmatrix} 0.94 \end{bmatrix}$$
$$V_i^{max} = \begin{bmatrix} 1.06 \end{bmatrix}$$
$$\theta_i = \begin{bmatrix} 0 & -4.98 \\ -12.72 & -10.33 \\ -8.78 & -14.22 \\ -13.37 & -13.36 \\ -14.94 & -15.1 \\ -14.79 & -15.07 \\ -15.16 & -16.04 \end{bmatrix}$$
$$\theta_{ij}^{min} = \begin{bmatrix} -360 \end{bmatrix}$$
$$\theta_{ij}^{max} = \begin{bmatrix} 360 \end{bmatrix}$$
$$G_{ij} = \begin{bmatrix} 0 \end{bmatrix}$$
$$B_{ij} = \begin{bmatrix} 0 \end{bmatrix}$$
$$P_g^{G,min} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$
$$P_g^{G,max} = \begin{bmatrix} 332.4 & 140 & 100 & 100 & 100 \end{bmatrix}$$
$$Q_g^{G,min} = \begin{bmatrix} 0 & -40 & 0 & -6 & -6 \end{bmatrix}$$
$$Q_g^{G,max} = \begin{bmatrix} 10 & 50 & 40 & 24 & 24 \end{bmatrix}$$
$$R_g^{G,down} = \begin{bmatrix} \infty \end{bmatrix}$$
$$R_g^{G,up} = \begin{bmatrix} \infty \end{bmatrix}$$

References

Amdahl, G.M., 1967. Validity of the single processor approach to achieving large scale computing capabilities, in: Proceedings of the April 18-20, 1967, spring joint computer conference, pp. 483–485.

Carpentier, J., 1962. Contribution a l'etude du dispatching economique. Bull. Soc. Fr. Elec. Ser. 3, 431.

Carpentier, J., 1979. Optimal power flows. International Journal of Electrical Power & Energy Systems 1, 3–15. URL: https://www.sciencedirect.com/science/article/pii/0142061579900267, doi:https://doi.org/10.1016/0142-0615(79)90026-7.

Chen, G., Qian, J., Zhang, Z., Sun, Z., 2019. Applications of novel hybrid bat algorithm with constrained pareto fuzzy dominant rule on multi-objective optimal power flow problems. IEEE Access 7, 52060–52084. doi:10.1109/ACCESS.2019.2912643.

Huang, S., Dinavahi, V., 2018a. Fast batched solution for real-time optimal power flow with penetration of renewable energy. IEEE Access 6, 13898–13910. doi:10.1109/ACCESS.2018.2812084.

Huang, S., Dinavahi, V., 2018b. Gpu-based parallel real-time volt/var optimisation for distribution network considering distributed generators. IET Generation, Transmission & Distribution 12, 4472–4481. doi:https://doi.org/10.1049/iet-gtd.2017.1887.

Li, X., Li, F., Yuan, H., Cui, H., Hu, Q., 2017. Gpu-based fast decoupled power flow with preconditioned iterative solver and inexact newton method. IEEE Transactions on Power Systems 32, 2695–2703. doi:10.1109/TPWRS.2016.2618889.

Liu, D.C., Nocedal, J., 1989. On the limited memory bfgs method for large scale optimization. Mathematical Programming 45, 503–528. URL: https://doi.org/10.1007/BF01589116, doi:10.1007/BF01589116.

Mohagheghi, E., Alramlawi, M., Gabash, A., Blaabjerg, F., Li, P., 2020. Real-time active-reactive optimal power flow with flexible operation of battery storage systems. Energies 13. URL: https://www.mdpi.com/1996-1073/13/7/1697, doi:10.3390/en13071697.

Mohagheghi, E., Alramlawi, M., Gabash, A., Li, P., 2018. A survey of real-time optimal power flow. Energies 11. URL: https://www.mdpi.com/1996-1073/11/11/3142, doi:10.3390/en11113142.

Mohagheghi, E., Gabash, A., Li, P., 2017. A framework for real-time optimal power flow under wind energy penetration. Energies 10. URL: https://www.mdpi.com/1996-1073/10/4/535, doi:10.3390/en10040535.

Oliveira, E.J., Oliveira, L.W., Pereira, J., Honório, L.M., Silva, I.C., Marcato, A., 2015. An optimal power flow based on safety barrier interior point method. International Journal of Electrical Power & Energy Systems 64, 977–985. URL: https://www.sciencedirect.com/science/article/pii/S0142061514005419, doi:https://doi.org/10.1016/j.ijepes.2014.08.015.

Rakai, L., Rosehart, W., 2014. Gpu-accelerated solutions to optimal power flow problems, in: 2014 47th Hawaii International Conference on System Sciences, pp. 2511–2516. doi:10.1109/HICSS.2014.315.

Reddy, S.S., 2017. Optimal power flow with renewable energy resources including storage. Electrical Engineering 99, 685–695. URL: https://doi.org/10.1007/s00202-016-0402-5, doi:10.1007/s00202-016-0402-5.

Roberge, V., Tarbouchi, M., Okou, F., 2016. Optimal power flow based on parallel metaheuristics for graphics processing units. Electric Power Systems Research 140, 344–353. URL: https://www.sciencedirect.com/science/article/pii/S0378779616302140, doi:https://doi.org/10.1016/j.epsr.2016.06.006.

Shin, S., Pacaud, F., Anitescu, M., 2024. Accelerating optimal power flow with gpus: Simd abstraction of nonlinear programs and condensed-space interior-point methods arXiv:2307.16830.

Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., Riedmiller, M., 2014. Deterministic policy gradient algorithms, in: Xing, E.P., Jebara, T. (Eds.), Proceedings of the 31st International Conference on Machine Learning, PMLR, Bejing, China. pp. 387–395. URL: https://proceedings.mlr.press/v32/silver14.html.

Su, X., He, C., Liu, T., Wu, L., 2020. Full parallel power flow solution: A gpu-cpu-based vectorization parallelization and sparse techniques for newton–raphson implementation. IEEE Transactions on Smart Grid 11, 1833–1844. doi:10.1109/TSG.2019.2943746.

Tang, Y., Dvijotham, K., Low, S., 2017. Real-time optimal power flow. IEEE Transactions on Smart Grid 8, 2963–2973. doi:10.1109/TSG.2017.2704922.

Wang, H., Murillo-Sanchez, C., Zimmerman, R., Thomas, R., 2007. On computational issues of market-based optimal power flow. Power Systems, IEEE Transactions on 22, 1185 – 1193. doi:10.1109/TPWRS.2007.901301.

Wang, Z., Wende-von Berg, S., Braun, M., 2021. Fast parallel newton–raphson power flow solver for large number of system calculations with cpu and gpu. Sustainable Energy, Grids and Networks 27, 100483. URL: https://www.sciencedirect.com/science/article/pii/S2352467721000540, doi:https://doi.org/10.1016/j.segan.2021.100483.

Woo, J., wu, L., Park, J.B., Roh, J., 2020. Real-time optimal power flow using twin delayed deep deterministic policy gradient algorithm. IEEE Access 8, 213611–213618. doi:10.1109/ACCESS.2020.3041007.

Wu, P., Chen, C., Lai, D., Zhong, J., 2023. A safe drl method for fast solution of real-time optimal power flow. arXiv:2308.03420.

Zimmerman, R.D., Murillo-Sánchez, C.E., Thomas, R.J., 2011a. Matpower: Steady-state operations, planning, and analysis tools for power systems research and education. IEEE Transactions on Power Systems 26, 12–19. doi:10.1109/TPWRS.2010.2051168.

Zimmerman, R.D., Murillo-Sánchez, C.E., Thomas, R.J., 2011b. Matpower: Steady-state operations, planning, and analysis tools for power systems research and education. IEEE Transactions on Power Systems 26, 12–19. doi:10.1109/TPWRS.2010.2051168.