

IET Generation, Transmission & Distribution

Special issue Call for Papers



**Be Seen. Be Cited.
Submit your work to a new
IET special issue**

**"Emerging Applications of
IoT and Cybersecurity for
Electrical Power Systems"**


**Lead Guest: Editor Mohamed
M. F. Darwish**

**Guest Editors: Mahmoud
Elsisi, Diao-Eldin A. Mansour,
Mostafa M. Fouda and Matti
Lehtonen**

Read more



Parallel primal-dual interior point method for the solution of dynamic optimal power flow

Rylee Sundermann¹  | Shrirang G. Abhyankar² | Hong Zhang³ | Jung-Han Kim¹ | Timothy M. Hansen⁴

¹Department of Mathematics and Statistics, South Dakota State University, Brookings, South Dakota, USA

²Electricity Infrastructure and Buildings Division, Pacific Northwest National Laboratory (PNNL), Richland, Washington, USA

³Computer Science Department, Illinois Institute of Technology, Chicago, Illinois, USA

⁴Department of Electrical Engineering and Computer Science, South Dakota State University, Brookings, South Dakota, USA

Correspondence

Rylee Sundermann, Department of Mathematics and Statistics, South Dakota State University, Brookings, SD 57007, USA.

Email: rylee.sundermann@jacks.sdstate.edu

Abstract

This work presents a novel solution for accelerating the dynamic optimal power flow using a distributed-memory parallelization approach. Unlike other two-stage relaxation-based approaches (such as ADMM), the proposed approach constructs the entire dynamic optimal power flow problem in parallel and solves it using a parallel primal-dual interior point method with an iterative Krylov subspace linear solver with a block-Jacobi preconditioning scheme. The parallel primal-dual interior point method has been implemented in the open-source portable, extensible toolkit for scientific computation (PETSc) library. The formulation, implementation, and numerical results on multicore computers to demonstrate the performance of the proposed approach on medium- to large-scale networks with varying time horizons are presented. The results show that a significant speedup is achieved by using a block-Jacobi preconditioner with an iterative Krylov subspace method for solving the dynamic optimal power flow problems.

1 | INTRODUCTION

The operation and planning of the electric power grid continues to increase in complexity with the advent of technologies promoting deeper decarbonization. The proliferation of renewable energy sources, such as wind and solar energy, continues to permeate through the grid, providing cleaner and economical sources of electricity. As their penetration increases, the role of storage and technologies for flexible resource utilization will be critical, and the analysis of their impact and economic benefits will be vital. However, the introduction of these technologies also increases the computational complexity of optimal operation, necessitating more accurate and faster solution methods.

The alternating current optimal power flow (ACOPF) [1, 2] is an important analysis tool for grid engineers and planners. Its aim is to find the optimal generation dispatch minimizing the production cost, while adhering to network and equipment operating limits. The dynamic optimal power flow (DOPF) problem, also referred to as multiperiod ACOPF, is an extension of ACOPF that adds the dimension of time. DOPF's aim is

to find the optimal generation dispatch over an operating time horizon while adhering to the network, equipment operating limits, and intertemporal constraints.

The complexity of DOPF with respect to computing and memory grows approximately linearly with the number of timesteps. Thus, the solution of large-scale DOPF problems over long time horizons is a challenging computational problem. Substantial work has been performed on DOPF [3–7], but all these approaches solve the DOPF problem sequentially, that is, on only one processing element. As a result, they do not scale to larger problems. Moreover, they ignore the near-independent structure of DOPF, thus causing a computational bottleneck when creating and updating the Karush–Kuhn–Tucker (KKT) system.

In [8–10], the authors distribute the system across multiple processors after the setup is done on a single processor. While this approach does improve the solution time, it is inefficient because it creates a communication bandwidth limitation as the single processor must distribute the information across the parallel compute system. Additionally, this leads to memory issues as the whole system must be able to fit in the memory of a

This is an open access article under the terms of the [Creative Commons Attribution-NonCommercial-NoDerivs License](https://creativecommons.org/licenses/by-nc-nd/4.0/), which permits use and distribution in any medium, provided the original work is properly cited, the use is non-commercial and no modifications or adaptations are made.

© 2022 The Authors. *IET Generation, Transmission & Distribution* published by John Wiley & Sons Ltd on behalf of The Institution of Engineering and Technology.

single processor (or compute node) prior to distribution. A similar parallel structure can be found in [11, 12], which utilizes the programming language Julia to handle the parallel calculations of the DOPF system and a parallel interior point method solver (PIPS). The authors solve a security-constrained optimal power flow problem using PIPS, which has the advantage of having a block-arrow matrix structure. Such a structure is very efficient for a Schur-complement decomposition approach. The DOPF problem results in a staircase matrix structure, however, which is not efficient for a Schur-complement decomposition approach.

The primal-dual interior point method (PDIPM) [13–15] solver used in this work is a generalization and can be used for all types of problem structures. The proposed block-Jacobi preconditioner removes the interprocessor communication that the LDL^T factorization requires. Moreover, the implementation in parallel allows for additional memory resources and lower memory requirement per processor. Our work is a direct expansion of [3], parallelizing ExaGO's TCOFLOW structure and creating an application interface with our PDIPM solver.

1.1 | Contributions

In this paper we present a novel approach to accelerate DOPF solutions to solve problems involving large-scale power systems with longer time-horizons. The main contributions of our work are as follows:

- (i) Development of a fully parallel PDIPM method for solving the DOPF problem with computational ease and efficiency, compared with existing approaches that solve the DOPF problem on a single processor (limiting application sizes) [4–7] or use decomposition approaches [8, 10] that implement sequential optimization steps with distributed execution of the KKT linear systems (limiting scalability).
- (ii) Design of an iterative Krylov-subspace method with a block-Jacobi preconditioner to overcome the memory bottleneck in solving internal linear systems. Our numerical experiments demonstrate its superiority and scalability on the large size DOPF problems.
- (iii) Release of an open-source parallel PDIPM implemented in the popular open-source library portable, extensible toolkit for scientific computation (PETSc) [16] to enable others to solve their applications, including problems arising from the operation and control of power generation, power system management, etc. on parallel computers, either for shorter development effort, faster computer execution time, or breaking memory barrier. While there are implementations of PDIPM for single-processor execution, IPOPT [17], or PIPS-IPM [18] for structured problems, the PDIPM implementation in PETSc is, to our knowledge, the first to solve general nonlinear optimization problems with constraints in parallel.

1.2 | Paper organization

The remaining sections of the paper are organized as follows. Section 2 describes the formulation of DOPF as an

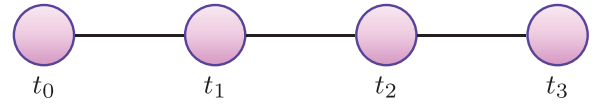


FIGURE 1 Multiperiod optimal power flow example with four time steps. The lines connecting the different time periods denote the coupling between them (adapted from [19])

optimization problem. In Section 3, the PDIPM for solving general nonlinear constrained optimization problems and its parallel implementation in PETSc are presented. Details about its application to DOPF are provided in Section 4. Section 5 presents the numerical simulation results and analysis of the parallel performance. Conclusions and future work are given in Section 6.

2 | FORMULATION OF DYNAMIC OPTIMAL POWER FLOW

The DOPF problem can be conceived as a series of ACOF problems connected with intertemporal constraints, as shown in Figure 1. Each timestep t_i is coupled with its preceding timestep $t_{i-1} = t_i - \Delta t$, and the objective is to find a least-cost dispatch for the given time horizon $T = [t_0, t_N]$. The equations for the general form of DOPF are given by Equations (1)–(5).

$$\min \sum_{t \in T} f(\mathbf{x}(t)) \quad (1)$$

s.t.

$$\mathbf{g}(\mathbf{x}(t)) = 0, \quad (2)$$

$$\mathbf{h}(\mathbf{x}(t)) \geq 0, \quad (3)$$

$$\mathbf{x}^- \leq \mathbf{x}(t) \leq \mathbf{x}^+, \quad (4)$$

$$-\mathbf{r}(t) \leq \mathbf{x}(t) - \mathbf{x}(t - \Delta t) \leq \mathbf{r}(t), \quad t \neq 0. \quad (5)$$

DOPF aims to minimize the total generation cost $\sum_{t \in T} f(\mathbf{x}(t))$ over the time horizon T . At each timestep, the equality constraints $\mathbf{g}(\mathbf{x}(t))$, inequality constraints $\mathbf{h}(\mathbf{x}(t))$, and lower/upper limit (\mathbf{x}^- , \mathbf{x}^+) constraints need to be satisfied. Equation (5) represents the coupling between the consecutive timesteps. In the rest of this section, we describe the different elements of the dynamic optimal power flow formulation including variables, objective function, constraints, and bounds.

2.1 | Variables

The solution vector $\mathbf{x}(t)$ consists of the generator real and reactive power and the bus voltages; $\mathbf{x}(t) = [P_{G_k}, Q_{G_k}, V_{R_k}, V_{I_k}]^T$. In this work, we use a Cartesian representation of the network voltages.

2.2 | Objective function

For minimizing power production across a set time horizon, $[0, T]$, the DOPF problem is decomposed into $N_t = T/\Delta t$ timesteps as described in [10, 20]. The objective of the DOPF problem is to minimize the total operating cost over the given time horizon and is expressed as follows

$$\sum_{t \in T} f(\mathbf{x}(t)) = \sum_{i \in T} \sum_{k=1}^{N_g} \left(\alpha_k P_{G_k}^2 + \beta_k P_{G_k} + \gamma_k \right), \quad (6)$$

where $\alpha_k, \beta_k, \gamma_k$ are the quadratic cost coefficients for generator k and N_g is the number of generators.

2.3 | Variable bounds

The generator power and bus voltages need to adhere to their operational limits, which are expressed as box bounds on the $\mathbf{x}(t)$ vector as follows:

$$\begin{bmatrix} P_G^- \\ Q_G^- \\ V_R^- \\ V_I^- \end{bmatrix} \leq \mathbf{x}(t) \leq \begin{bmatrix} P_G^+ \\ Q_G^+ \\ V_R^+ \\ V_I^+ \end{bmatrix} \quad (7)$$

where $P_G^-, P_G^+, Q_G^-,$ and Q_G^+ are given and generator dependent, and bus voltage is unbounded $V_R^-, V_I^- = -\infty$ and $V_R^+, V_I^+ = \infty$.

2.4 | Equality constraints

The nodal power balance constraints at each time-step form the equality constraints for the problem and are given by:

$$\begin{aligned} \Delta P_f &= \sum_{A_{br}(f,t)=1} (G_{ff}(V_{Rf}^2 + V_{If}^2) + V_{Rf}(G_{\beta} V_{Rt} \\ &\quad - B_{\beta} V_{It}) + V_{If}(B_{\beta} V_{Rt} + G_{\beta} V_{It})) \\ &\quad - \sum_{A_G(f,k)=1} P_{Gk} + \sum_{A_L(f,j) \neq 0} (P_{Dj}) = 0, \end{aligned} \quad (8)$$

$$\begin{aligned} \Delta Q_f &= \sum_{A_{br}(f,t)=1} (-B_{ff}(V_{Rf}^2 + V_{If}^2) + V_{If}(G_{\beta} V_{Rt} \\ &\quad - B_{\beta} V_{It}) - V_{Rf}(B_{\beta} V_{Rt} + G_{\beta} V_{It})) \\ &\quad - \sum_{A_G(f,k) \neq 0} Q_{Gk} + \sum_{A_L(f,j)=1} (Q_{Dj}) = 0, \end{aligned} \quad (9)$$

$$\Delta \theta_{ref} = V_{Iref} - V_{Rref} \tan(\theta_{ref}) = 0, \quad (10)$$

where Equations (8) and (9) are the real and reactive power balance equations, respectively, and Equation (10) holds the

reference voltage angle constant. A_{br}, A_G, A_L represent sparse incidence matrices mapping the lines, generators, and loads to buses, respectively. $A_{br}(f, t) = 1$ indicates a branch connection between buses f (from) and t (to). Similarly, $A_G(f, k) = 1$ indicates generator f is incident at bus k , and $A_L(f, j) = 1$ indicates load f is incident at bus j .

2.5 | Inequality constraints

The inequality constraints are the flow limits (Equation (11)) for the transmission lines and voltage magnitude bounds (Equation (12)) at each time-step

$$0 \leq \left(P_{ft}^2 + Q_{ft}^2 \right) \leq (S_{ft}^+)^2, \quad (11)$$

$$(V_{min})^2 \leq V_{Ri}^2 + V_{Li}^2 \leq (V_{max})^2. \quad (12)$$

When considering the power flow between buses, S_{ft}^+ can vary as each transmission line is rated for normal, short-term, and emergency operation, labelled RATE_A, RATE_B, RATE_C, respectively. In this paper, we use RATE_A rating of lines.

The other set of inequality constraints arises from the ramp rate limits for generators between consecutive time-periods, that is, every generator output can ramp up or down within its ramp-rate capability. This can be expressed through Equations (13) and (14).

$$P_G(t - \Delta t) - P_G(t) \leq r_{Gk} \Delta t, \quad (13)$$

$$P_G(t) - P_G(t - \Delta t) \leq r_{Gk} \Delta t. \quad (14)$$

3 | A PARALLEL PRIMAL-DUAL INTERIOR POINT METHOD IN PETSC

In this section we construct a PDIPM for solving constrained nonlinear optimization problems. Consider the described optimization problem for DOPF, the objective function $f(\mathbf{x})$, equality constraints $\mathbf{g}(\mathbf{x})$, inequality constraints $\mathbf{h}(\mathbf{x})$, and upper and lower bounds \mathbf{x}^- and \mathbf{x}^+ on \mathbf{x} , in general. Then an optimization problem is written in compact form as

$$\begin{aligned} \min_{\mathbf{x}} \quad & f(\mathbf{x}) \\ \text{s.t.} \quad & \mathbf{g}(\mathbf{x}) = 0, \\ & \mathbf{h}(\mathbf{x}) \geq 0, \\ & \mathbf{x}^- \leq \mathbf{x} \leq \mathbf{x}^+. \end{aligned} \quad (15)$$

From Equation (15), we combine the constraints and the bounds to obtain

$$G(\mathbf{x}) = \begin{bmatrix} \mathbf{g}(\mathbf{x}) \\ \mathbf{x} - \mathbf{x}_{eq} \end{bmatrix} \text{ and } H(\mathbf{x}) = \begin{bmatrix} \mathbf{h}(\mathbf{x}) \\ \mathbf{x}^+ - \mathbf{x} \\ \mathbf{x} - \mathbf{x}^- \end{bmatrix},$$

where \mathbf{x}_{eq} represents the set of \mathbf{x} variables where $\mathbf{x}^- = \mathbf{x}^+$. Introducing a set of slack variables $\boldsymbol{\zeta}$ and a logarithmic barrier to ensure the positivity of $\boldsymbol{\zeta}$, then Equation (15) is formulated as a new optimization problem:

$$\begin{aligned} \min_{\mathbf{x}} \quad & f(\mathbf{x}) - \mu \sum_{i=1}^{N_H} \ln(\zeta_i) \\ \text{s.t.} \quad & \mathbf{G}(\mathbf{x}) = \mathbf{0}, \\ & \mathbf{H}(\mathbf{x}) - \boldsymbol{\zeta} = \mathbf{0}, \end{aligned} \quad (16)$$

where N_H denotes the number of inequality constraints in H . We note that μ is driven to zero during the optimization.

Define $\mathbf{X} = [\mathbf{x}, \boldsymbol{\lambda}_G, \boldsymbol{\lambda}_H, \boldsymbol{\zeta}]^T$. Then the final transformation of Equation (16) is a single Lagrangian function [21]:

$$L_\mu(\mathbf{X}) = f(\mathbf{x}) + \boldsymbol{\lambda}_G^T \mathbf{G}(\mathbf{x}) - \boldsymbol{\lambda}_H^T (\mathbf{H}(\mathbf{x}) - \boldsymbol{\zeta}) - \mu \sum_{i=1}^{N_H} \ln \zeta_i, \quad (17)$$

where $\boldsymbol{\lambda}_G$ and $\boldsymbol{\lambda}_H$ are Lagrangian multipliers for the equality and inequality constraints, respectively. Additionally, the solution to Equation (16) will result in a saddle point in Equation (17). Applying Newton's method with Equation (17) as the target function, we aim to find a critical point \mathbf{x}^* that will solve our original optimization problem in Equation (15).

The minimizer \mathbf{X}^* of Equation (17) must satisfy KKT conditions [22], which allow an extension of Lagrangian multipliers to inequality constraints. We apply a Newton's method to refine an initial guess \mathbf{X}_0 by $\mathbf{X}_{n+1} = \mathbf{X}_n + \alpha \Delta \mathbf{X}$, in which the search direction $\Delta \mathbf{X}$ is calculated by solving

$$\mathbf{K} \Delta \mathbf{X} = -\mathbf{F}, \quad (18)$$

the symmetric KKT matrix $\mathbf{K} = \nabla^2 L_\mu$ is

$$\begin{bmatrix} \mathbf{W}_{xx} & \nabla \mathbf{G}(\mathbf{x})^T & -\nabla \mathbf{H}(\mathbf{x})^T & \mathbf{0} \\ \nabla \mathbf{G}(\mathbf{x}) & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ -\nabla \mathbf{H}(\mathbf{x}) & \mathbf{0} & \mathbf{0} & \mathbf{I} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} & \boldsymbol{\Lambda}_H * \mathbf{Z}^{-1} \end{bmatrix}, \quad (19)$$

the right-hand side vector

$$\mathbf{F} = \begin{bmatrix} \mathbf{W}_x \\ \mathbf{G}(\mathbf{x}) \\ \boldsymbol{\zeta} - \mathbf{H}(\mathbf{x}) \\ \boldsymbol{\Lambda}_H \mathbf{e} - \mu * \mathbf{Z}^{-1} \mathbf{e} \end{bmatrix}, \quad (20)$$

$$\mathbf{W}_x = \nabla f(\mathbf{x})^T + \nabla \mathbf{G}(\mathbf{x})^T \boldsymbol{\lambda}_G - \nabla \mathbf{H}(\mathbf{x})^T \boldsymbol{\lambda}_H, \quad (21)$$

$$\mathbf{W}_{xx} = \nabla^2 f(\mathbf{x}) + \nabla^2 \mathbf{G}(\mathbf{x})^T \boldsymbol{\lambda}_G - \nabla^2 \mathbf{H}(\mathbf{x})^T \boldsymbol{\lambda}_H, \quad (22)$$

where \mathbf{e} is a vector of ones, \mathbf{I} is the identity matrix, and \mathbf{Z} and $\boldsymbol{\Lambda}_H$ are square matrices with $\boldsymbol{\zeta}$ and $\boldsymbol{\lambda}_H$ along their diagonals, respectively.

The KKT matrix (Equation (19)) would become more indefinite and ill-conditioned for large-scale problems. The sequential PDIPM software packages (e.g. IPOPT and MIPS) use a reduced KKT matrix to improve computational efficiency. However, evaluation of the reduced KKT matrix requires parallel matrix–matrix products that incur significant data movement between processors and lead to a denser submatrix \mathbf{W}_{xx} . Thus, we use the uncompressed KKT matrix (Equation (19)) in our current PDIPM/PETSc implementation.

To ensure a descent direction $\Delta \mathbf{X}$, we expect no zero inertia indices and the number of primal and dual variables to match the number of positive and negative inertia indices, respectively. We apply an \mathbf{LDL}^T [23] matrix factorization preconditioner provided by the latest release of Multifrontal Massively Parallel sparse direct Solver (MUMPS-v5.4.1) library [24] to \mathbf{K} and evaluate its matrix inertia [25]. Should there be a different distribution of the matrix inertia, we introduce shift δ_w to balance positive and negative inertia indices and add shift δ_c to remove zero indices. We adopt the same heuristics and algorithm for shifts as IPOPT. This formulation can be represented as

$$\begin{bmatrix} \mathbf{W}_{xx} + \delta_w * \mathbf{I} & \nabla \mathbf{G}(\mathbf{x})^T & -\nabla \mathbf{H}(\mathbf{x})^T & \mathbf{0} \\ \nabla \mathbf{G}(\mathbf{x}) & -\delta_c * \mathbf{I} & \mathbf{0} & \mathbf{0} \\ -\nabla \mathbf{H}(\mathbf{x}) & \mathbf{0} & -\delta_c * \mathbf{I} & \mathbf{I} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} & \boldsymbol{\Lambda}_H * \mathbf{Z}^{-1} \end{bmatrix}. \quad (23)$$

The evaluation of the matrix inertia counts the number of positive, negative and zero diagonals of the numerically factored matrix. It costs ignorable computation and communication time compared to the numerical factorization of the matrix. This process is implemented in MUMPS.

At the end of each iteration, we test the convergence of the Newton's solver by testing the norm of \mathbf{F} separated between prime and dual components described as the residual norm

$$r = \sqrt{\mathbf{W}_x^T \mathbf{W}_x + (\mathbf{Z} \boldsymbol{\Lambda}_H \mathbf{e} - \mu \mathbf{e})^T (\mathbf{Z} \boldsymbol{\Lambda}_H \mathbf{e} - \mu \mathbf{e})} \quad (24)$$

and constraint norm (c-norm)

$$\tau = \sqrt{\mathbf{G}(\mathbf{x})^T \mathbf{G}(\mathbf{x}) + (\boldsymbol{\zeta} - \mathbf{H}(\mathbf{x}))^T (\boldsymbol{\zeta} - \mathbf{H}(\mathbf{x}))}. \quad (25)$$

These norms are then used as convergence criteria: residual convergence and constraint convergence. The convergence is reached when the c-norm and either absolute or relative residual tolerances are met. These criteria are based on the convergence of primal, dual, and complementary slackness as used in most optimization solvers, for example, [17].

The PDIPM is implemented as a general constrained optimization solver in PETSc. Appendix A illustrates its software design, implementation, and usage.

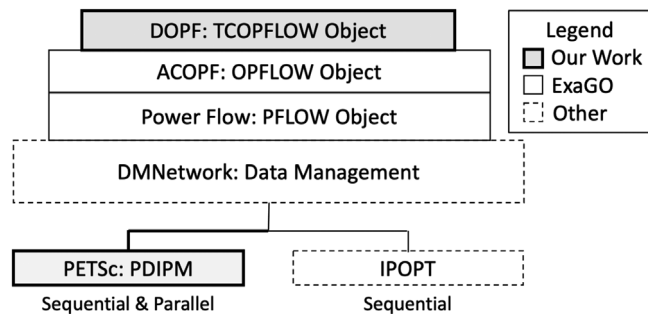


FIGURE 2 Software structure of DOPF within ExaGO

4 | APPLICATION OF PDIPM to DOPF

The DOPF application is built on top of the ExaGO framework [3]. It utilizes the DMNetwork foundation [26] in PETSc to handle the construction and parallel distribution of the underlying power network, data movement, and ACOPF objects (see OPFLOW in [3]). Figure 2 illustrates the software structure of DOPF using ExaGO, as well as how this work and parallel PDIPM add new extensions to the ExaGO and PETSc libraries.

We note here that we solve the entire DOPF (including all the time-periods) as a single problem. We do not decouple the equations and solve them using two-stage approaches such as ADMM. The equations for each time-period are assigned to individual processors and there are equations that couple variables across different processors to model the dependency between time-step t_i and t_{i+1} . So, in essence the DOPF model is partitioned across many processors and solved in parallel using the PDIPM solver developed in PETSc.

The full optimization system in Equation (15) is solved by using PDIPM/PETSc as presented in the preceding section. The internal nonlinear algebraic equations (Equations (17) and (18)) are solved via Newton–Krylov methods by using PETSc SNES [27].

In general, the solutions of linear systems dominate the computation. In this section, we discuss the data structures of the KKT matrices arising from the DOPF application, based on which we present two preconditioners for solving the KKT systems (Equation (18)).

Observing the full optimization system (Equation (15)), we see that the only link between the variables associated with the timesteps is the ramping constraints in Equations (13) and (14). If the system is ordered such that the variables from the first time period appear first, then the variables from the next timestep, the resulting KKT matrix has an interesting blocked-diagonal structure that allows minimal communication between processors in the computation. Therefore, we distribute DOPF as a full set of ACOPF systems, where each processor contains at least one timestep.

For example, consider simulating a DOPF with a half-hour time period where the load data has resolution of 5 min. For our purposes, this equates to $N_t = \frac{30 \text{ min}}{5 \text{ min}} = 6$ timesteps. Figure 3a shows the data structure of the resulting KKT matrix on a single processor. If we distribute this to 2 processors, we would have three timesteps per processor with processor 1 holding

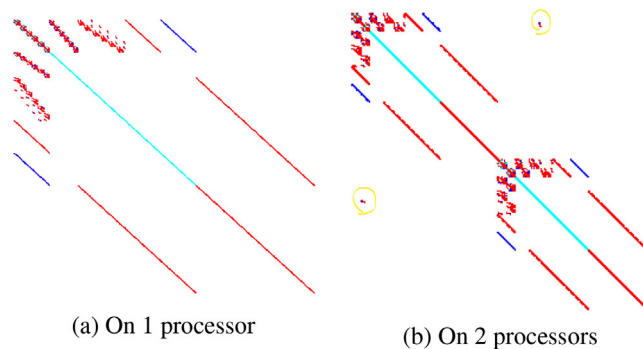


FIGURE 3 KKT matrix structure with six timesteps

$t = 1, 2, 3$ and processor 2 holding 4, 5, 6 timesteps, as shown in Figure 3b. Having each timestep represented as an independent power network, we can initialize our DOPF solution by solving ACOPF separately on each timestep independently in parallel.

From here, the network constraints described in Equation (15) can be solved in parallel, running the DOPF update routines mentioned earlier. Additionally, the time constraints can be computed without communication if $t - 1$ and t from Equations (13) and (14) are on the same processor. In this example, the ramp conditions linking timesteps 1 & 2, 2 & 3, 4 & 5, and 5 & 6 are handled without communication, while those linking 3 & 4 require minimal local communication as indicated by Figure 3b.

With the formulation of the KKT system, at each iteration of Newton’s method, a linear solve (i.e., a Krylov subspace method) is required to find the search direction $\Delta \mathbf{X}$ from Equation (18). The indefinite and highly ill-conditioned nature of KKT matrices requires a robust preconditioner. We apply an \mathbf{LDL}^T direct solver and the inertia correction method described in Equation (23) to aid convergence.

As the system (Equation (15)) becomes large when the timesteps increase, the \mathbf{LDL}^T direct solver consumes too much memory and/or reduces the parallel scalability. We then take advantage of the block-diagonal structure of the KKT matrices and apply the block-Jacobi preconditioner [27] with \mathbf{LDL}^T matrix factorization applied to the inner subblocks, as shown by Figure 4.

Our numerical experiments show that the block-Jacobi preconditioner yields better parallel speedups although it takes more total PDIPM iterations to converge. PETSc library provides numerous options to allow users to select solvers and algorithms at runtime. When the DOPF problem is modified or the KKT matrix structure changes, users can experiment with various options and find a feasible and efficient solver. Appendix B presents the options used in our numerical experiments.

5 | NUMERICAL EXPERIMENTS

Two standard power systems from the Texas A&M Synthetic Test Case Repository, 200-Bus and 2000-Bus systems [28, 29],

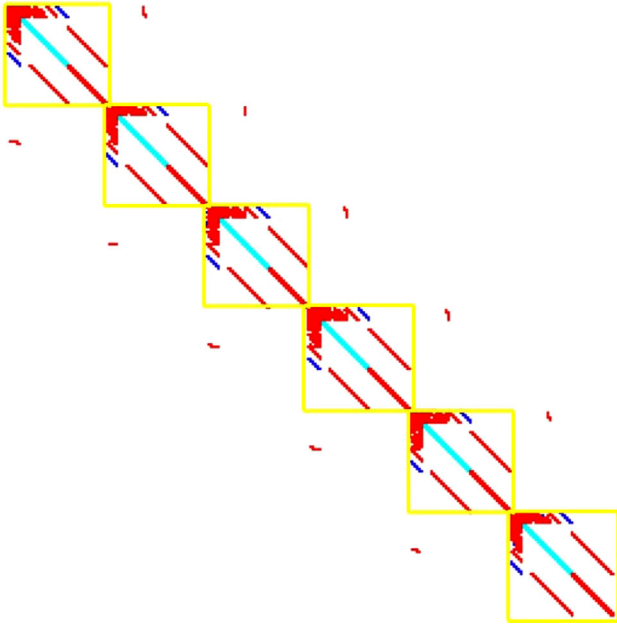


FIGURE 4 KKT matrix structure of 200-bus system with six timesteps on six processors

were used to evaluate our PDIPM implementation in PETSc. For DOPF results, we utilized variable load demands on the 200-Bus system to demonstrate viability on real-world data[3] and two different load profiles on the 2000-Bus system for scope and scalability. The different load profiles were designed to show the potential of the solver within $\delta_t \mathbf{x}$ on a large system (Tables 3 and 4) and test performance when tightly linked from inter-temporal constraints (Table 5). We load data every 5 min for the 200-Bus system and every one hour for the 2000-Bus system; thus the number of timesteps $N_t = \text{Duration (min)}/\text{Time Interval (min)}$. For parallel computation, we set the number of processor cores $N_p = N_t$; in other words, each core holds an independent power network subsystem at a single timestep, as discussed in Section 4.

Our DOPF is built on the ExaGO framework [3], which consists of ACOF and PFLOW objects, and utilizes the DMNetwork class [26] in PETSc to handle the construction and parallel distribution of the underlying power network, as shown in Figure 2. Our DOPF code first reads network data for a single timestep, as shown in Figure 5. Using DMNetwork/PETSc API functions, we created the power network; registered network components, such as transmission lines as edges, buses and generators as vertices; added these components to the network; and then DMNetwork assembled and distributed the resulting network to all processor cores [30]. The linear, nonlinear, and optimization solvers were built on the top of this network via standard PDIPM/PETSc API functions.

We conducted experiments on two computer systems: (i) a Linux server and (ii) Theta, an Intel-Cray XC40 system in the Argonne Leadership Computing Facility [31]. The Linux server has dual Intel Xeon Gold 6130 CPUs at 2.1 GHz with 32 cores (64 threads) and 192 GiB of RAM. Theta has 4392 nodes, each

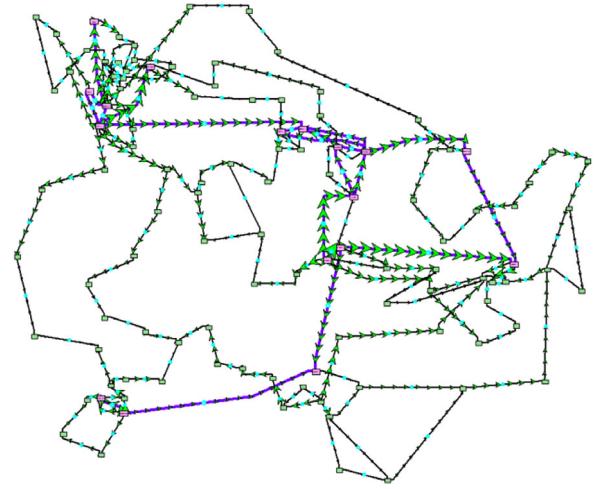


FIGURE 5 ACTIVSg200 network for a single timestep (adapted from [28])

TABLE 1 Total time of 200-Bus system on Theta (s)

Duration	N_t	N_{Var}	\mathbf{LDL}^T		b-Jacobi $N_p = N_t$	Speedup
			$N_p = 1$	$N_p = N_t$		
0.5 h	6	25,726	216	150	56	3.9
1 h	12	51,604	658	608	88	7.5
2 h	24	103,360	2043	972	135	15.1
4 h	48	206,872	6767	2704	476	14.2

N_{Var} : Number of variables in \mathbf{x} . N_p : Number of cores. Speedup: Column 4 / Column 6.

TABLE 2 Total time of 200-Bus system on the Linux server (s)

Duration	N_t	IPOPT $N_p = 1$	\mathbf{LDL}^T		b-Jacobi $N_p = N_t$	Speedup
			$N_p = 1$	$N_p = N_t$		
0.5 h	6	13	19	15	6.3	3.0
1 h	12	35	57	67	14	4.1
2 h	24	102	187	108	16	11.7
4 h	48	581	621	520	87	7.1

N_p : Number of cores. Speedup: Column 4 / Column 6.

with 64 1.3 GHz Intel Xeon Phi 7230 cores with 16 GiB of MCDRAM per node.

Tables 1 and 2 show the total execution time of a 200-Bus system on the Theta supercomputer and on the Linux server. Tables 3 and 5 present experimental results of the 2000-Bus system. More than 90% of the computation is spent on solving the KKT linear systems (Equation (18)), for which we compare three preconditioners used in the generalized minimal residual (GMRES) Krylov subspace iterations: sequential \mathbf{LDL}^T ($N_p = 1$), parallel \mathbf{LDL}^T ($N_p = N_t$), and block-Jacobi using N_p diagonal blocks with sequential \mathbf{LDL}^T applied to each inner sub-block of the KKT matrix (b-Jacobi).

TABLE 3 Total time of 2000-Bus system (first load profile) on the Linux server (s)

Duration	N_t	N_{var}	LDL^T		b-Jacobi $N_p = N_t$
			$N_p = 1$	$N_p = N_t$	
2 h	2	34,554	136	143	163
10 h	10	172,770	Out of Mem.	693	592
20 h	20	345,540	Out of Mem.	Out of Mem.	3192

N_{var} : Number of variables in X . N_p : Number of cores. Out of Mem.: Insufficient memory during matrix factorization.

TABLE 4 Total time of 2000-Bus system (first load profile) on Theta (s)

Duration	N_t	LDL^T		b-Jacobi $N_p = N_t$
		$N_p = 1$	$N_p = N_t$	
10 h	10	10,670	4301	5849
20 h	20	Out of Mem.	Out of Mem.	20,680

N_p : Number of cores. Out of Mem.: Insufficient memory during matrix factorization.

TABLE 5 Total time of 2000-Bus system (second load profile) on Linux server (s)

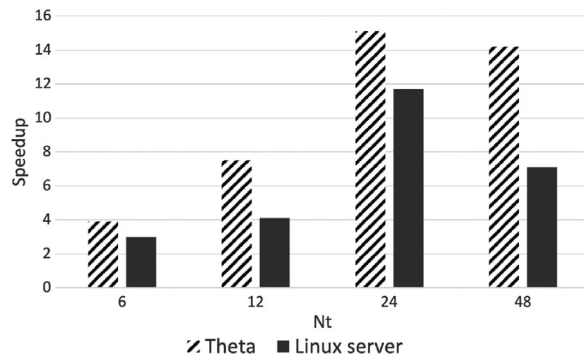
Duration	N_t	LDL^T		b-Jacobi $N_p = N_t$
		$N_p = 1$	$N_p = N_t$	
2 h	2	251	244	1034
10 h	10	Out of Mem.	Out of Mem.	17,223

N_p : Number of cores. Out of Mem.: Insufficient memory during matrix factorization.

PETSc library enables users to experiment various solver options at runtime without modifying application code. The options we used for the MUMPS' LDL^T preconditioner and block-Jacobi preconditioner with LDL^T applied to the inner sub-block are shown in Appendix B.

We utilized IPOPT via ExaGO to validate the accuracy of our PDIPM. For all the test systems, our PDIPM solutions gave the numerically identical optimal values of the objective function $f(\mathbf{x}^*(t))$ as IPOPT under the same convergence criteria presented in Section 3. As a reference, Table 2 lists the numerical performance of IPOPT that is comparable with the results of LDL^T preconditioner, but performs worse than our parallel PDIPM with the block-Jacobi preconditioner.

A subsystem of the 200-Bus system at a given timestep has ≈ 4287 variables, while the 2000-Bus system consists of 17,277 variables in each subsystem (i.e. $\approx 4\times$ larger). These numbers determine the size of the diagonal blocks of the KKT matrix, as shown in Figure 4. Although the 200-Bus system with 4-h duration has more total variables in X than the 2000-Bus system with 10-h duration, its KKT matrix consists of 48 small diagonal blocks compared with ten larger and denser diagonal blocks for the 2000-Bus system. The latter requires much larger memory for LDL^T matrix factorization.

**FIGURE 6** Speedup of block-Jacobi preconditioner on 200-Bus system on the Linux server and Theta ALCF machine

For small- to medium-size power systems (e.g., 0.5- and 1-h duration of 200-Bus system), parallel LDL^T does not show noticeable advantages over the sequential runs. As the size of the problems increases (e.g., 10- and 20-h duration of 2000-Bus system) sequential runs fail due to insufficient memory during LDL^T matrix factorization, while the parallel PDIPM with appropriate preconditioners successfully computes solutions. In most experiments, the block-Jacobi preconditioner outperforms LDL^T and gives speedups over the sequential LDL^T ranging from 3.0 to 15.1, as shown in Figure 6.

The first load profile gives loosely linked inter-temporal constraints, while the second load profile produces tighter links. For the latter, the block-Jacobi preconditioner becomes less efficient, requiring far more inner linear iterations and longer execution time as shown in Table 5. We conclude that the block-Jacobi preconditioner improves parallel performance on certain systems. For large size power systems, the primary benefit is its ability to converge to the optimal solution with much less memory usage compared to the large memory overhead in LDL^T factorization.

In addition to the solutions of the KKT systems, some researchers found the calculation of gradients and the Hessian matrices to be the next most computationally expensive aspect [4]. We calculate the analytic derivatives in the sparse gradients and Hessian matrix at each timestep simultaneously across multiple processors via ExaGO's ACOPF framework. Our experiments show that the gradient and Hessian evaluations took less than 5% of the total computation time and achieved superlinear speedup on almost all parallel tests. Table 6 presents the total time spent on the Hessian evaluations on Theta using the LDL^T preconditioner. The superlinear speedup

TABLE 6 Hessian evaluations on Theta

Test system	N_t	Total time (s)		Speedup
		$N_p = 1$	$N_p = N_t$	
200-Bus	48	118	1.6	73.75
2000-Bus*	10	364	16	22.8

N_p : Number of cores. Speedup: Column 3 / Column 4. *: First load profile.

likely comes from the cache performance. The performance of 200-Bus is significant because it would have a high cache hit ratio [32].

6 | CONCLUSIONS AND FUTURE WORK

This work presents solution of dynamic optimal power flow with the following innovations:

- (1) Instead of using conventional relaxation-based approaches, such as ADMM, we solved the entire DOPF problem using a fully parallel primal-dual interior point method (PDIPM),
- (2) A block-Jacobi preconditioner is used to overcome the memory bottleneck in solving KKT linear systems and thereby maximizing parallel performance. Our analysis and numerical results demonstrate a significant speedup on large-scale networks with varying time horizons, and
- (3) the parallel PDIPM solver is available publicly to benefit other researchers in their endeavors. To our knowledge, this is the first freely available parallel PDIPM for general nonlinear optimization problems with constraints.

Future work will consider expanding DOPF model to include energy storage (additional inter-temporal constraints) and use wind and other renewable energy as stochastic variables towards a stochastic-DOPF. In addition, scalable and robust solvers for the KKT linear systems for large-scale DOPF problems will be investigated.

NOMENCLATURE

S_f, S_l	Apparent power flow at from and to ends of line
$\alpha_k, \beta_k, \gamma_k$	Generator cost coefficients
\cdot_{ff}, \cdot_{ft}	Line link from-from and from-to
N_f	Number of timesteps
N_g	Number of generators
$\delta_{i,x}$	Ramp capability
r_{Gk}	Ramping capability of generator k
P_{Gk}, Q_{Gk}	Real and reactive power at generator k
$\Delta P_f, \Delta Q_f$	Real and reactive power balance
θ_{ref}	Reference bus voltage angle
V_{Rref}, V_{Iref}	Real and imaginary components of voltage at the reference voltage
G_{ff}, G_{ft}	Self and mutual conductance
B_{ff}, B_{ft}	Self and mutual susceptance
\cdot^+, \cdot^-	Upper and lower bounds on a variable
V_i, V_{Ri}, V_{Ii}	Voltage magnitude, real, and imaginary voltage components at bus i

AUTHOR CONTRIBUTIONS

Rylee Sundermann: Conceptualization, methodology, software, validation, writing - original draft, writing - review and editing. Shrirang Abhyankar: Methodology, software, supervision, writing - review and editing. Hong Zhang: Methodology, software, supervision, writing - review and editing. Jung-Han Kimn:

Methodology, supervision, writing - review and editing. Timothy Hansen: Conceptualization, methodology, supervision, writing - review and editing.

ACKNOWLEDGEMENTS

The authors thank Alp Dener for insightful discussions and Satish Balay and Jed Brown for assistance utilizing the Theta ALCF machine. This material is based upon work supported in part by the U.S. Department of Energy Office of Science, under contract DE-AC02-06CH11357.

CONFLICT OF INTEREST

The authors have declared no conflict of interest.

DATA AVAILABILITY STATEMENT

The real world data that support the findings of this study are openly available in ExaGO at <https://gitlab.pnnl.gov/exasgd/frameworks/exago>, reference number [4].

ORCID

Rylee Sundermann  <https://orcid.org/0000-0003-2161-088X>

REFERENCES

1. Carpentier, J.: Optimal power flows. *Int. J. Electr. Power Energy Syst.* 1(1), 3–15 (1979)
2. O'Neill, R.P., Castillo, A., Cain, M.B.: The IV Formulation and Linear Approximations of the AC Optimal Power Flow problem (OPF Paper 2). FERC Staff Technical Paper, pp. 1–18 (2012) [Online]. <http://www.ferc.gov/industries/electric/indus-act/market-planning/opf-papers/acopf-2-iv-linearization.pdf>
3. Abhyankar, S., Peles, S., Mancinelli, A., Rutherford, C.: Exago GIT Repository. <https://gitlab.pnnl.gov/exasgd/frameworks/exago>
4. Zaferanlouei, S., Farahmand, H., Vadlamudi, V.V., Korpas, M.: Battpower Toolbox: Memory-efficient and high-performance multi-period AC optimal power flow solver. *IEEE Trans. Power Syst.* 36(5), (2021)
5. Zaferanlouei, S., Korpas, M., Aghaei, J., Farahmand, H., Hashemipour, N.: Computational efficiency assessment of multi-period AC optimal power flow including energy storage systems. In: *International Conference on Smart Energy Systems and Technologies*, pp. 1–6. IEEE, Piscataway, NJ (2018)
6. Gill, S., Kockar, I., Ault, G.W.: Dynamic optimal power flow for active distribution networks. *IEEE Trans. Power Syst.* 29(1), 121–131 (2014)
7. Tang, C., Xu, J., Sun, Y., Liao, S., Ke, D., Li, X.: Stochastic dynamic optimal power flow in distribution network with distributed renewable energy and battery energy storage. *arXiv:1706.09995* (2017)
8. Mosaddegh, A., Canizares, C.A., Bhattacharya, K.: Distributed computing approach to solve unbalanced three-phase DOPFs. *IEEE Electrical Power and Energy Conference*, pp. 408–413. IEEE, Piscataway, NJ (2015)
9. Gerstner, P., Schick, M., Heuveline, V., Meyer-h, N., Leibfried, T., Slednev, V., Fichtner, W., Bertsch, V.: A domain decomposition approach for solving dynamic optimal power flow problems in parallel with application to the German transmission grid. EMCL preprint series, Heidelberg University, Heidelberg, (2016)
10. Schween, N., Gerstner, P., Meyer-Hübner, N., Slednev, V., Leibfried, T., Fichtner, W., Bertsch, V., Heuveline, V.: A domain decomposition approach to solve dynamic optimal power flow problems in parallel. In: Bertsch, V., Ardane, A., Suriyah, M., Fichtner, W., Leibfried, T., Heuveline, V. (eds.) *Advances in Energy System Optimization*, pp. 41–64. Birkhäuser, Cham (2018)
11. Schanen, M., Gilbert, F., Petra, C.G., Anitescu, M.: Toward multiperiod AC-based contingency constrained optimal power flow at large scale. 2018 Power System Computation Conference, pp. 1–7. IEEE, Piscataway, NJ (2018)

12. Rao, V., Kim, K., Schanen, M., Maldonado, D.A., Petra, C., Anitescu, M.: A multiperiod optimization-based metric of grid resilience. IEEE Power and Energy Society General Meeting, pp. 1–5. IEEE, Piscataway, NJ (2019)
13. Gay, D., Overton, M., Wright, M.: A primal-dual interior method for nonconvex nonlinear programming. In: Yuan, Y.X. (eds) Advances in Nonlinear Programming. Applied Optimization, vol. 14. Springer, Boston, MA, (1998)
14. Potra, F., Wright, S.: Interior-point methods. J. Comput. Appl. Math. 124(1), 281–302 (2000). <https://www.sciencedirect.com/science/article/pii/S0377042700004337>
15. Wright, M.: The interior-point revolution in optimization: history, recent developments, and lasting consequences. Bull. Amer. Math. Soc. (N.S.) 42, 39–56 (2005)
16. Balay, S., Abhyankar, S., Adams, M.F., Brown, J., Brune, P., Buschelman, K., Dalcin, L., Dener, A., Eijkhout, V., Gropp, W.D., Karpeyev, D., Kaushik, D., Knepley, M.G., May, D.A., McInnes, L.C., Mills, R.T., Munson, T., Rupp, K., Sanan, P., Smith, B.F., Zampini, S., Zhang, H., Zhang, H.: PETSc Web page. <https://www.mcs.anl.gov/petsc>, 2019. doi: <https://www.mcs.anl.gov/petsc>
17. Waechter, A., Biegler, L.T.: On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming. Math. Program. 106(1), 25–57 (2006)
18. Petra, C.G., Schenk, O., Anitescu, M.: Real-time stochastic optimization of complex energy systems on high-performance computers. Comput. Sci. Eng. 16(5), 32–42 (2014)
19. Abhyankar, S., Peles, S., Mancinelli, A., Rutherford, C.: ExaGO Manual version 1.0. doi: <https://gitlab.pnnl.gov/exasgd/frameworks/exago/-/blob/master/docs/manual/manual.pdf>
20. Meyer-Huebner, N., Suriyah, M., Leibried, T.: On efficient computation of time constrained optimal power flow in rectangular form. IEEE Eindhoven PowerTech, pp. 1–6. IEEE, Piscataway, NJ (2015)
21. Lagrange multiplier. doi: https://en.wikipedia.org/wiki/Lagrange_multiplier
22. Karush Kuhn Tucker conditions. doi: https://en.wikipedia.org/wiki/Karush%E2%80%93Kuhn%E2%80%93Tucker_conditions
23. Cholesky decomposition. doi: https://en.wikipedia.org/wiki/Cholesky_decomposition#LDL_decomposition
24. Amestoy, P., Buttari, A., L'Excellent, J.-Y., Mary, T.: Performance and scalability of the block low-rank multifrontal factorization on multicore architectures. ACM Trans. Math. Software 45, 2:1–2:26 (2019)
25. Sylvester's law of inertia. doi: https://en.wikipedia.org/wiki/Sylvester%27s_law_of_inertia
26. Abhyankar, S., Betrie, G., Maldonado, D., McInnes, L., Smith, B., Zhang, H.: PETSc DMNetwork: A library for scalable network pde-based multiphysics simulations. ACM Trans. Math. Software 46(1), (2020)
27. Balay, S., Abhyankar, S., Adams, M.F., Brown, J., Brune, P., Buschelman, K., Dalcin, L., Dener, A., Eijkhout, V., Gropp, W.D., Karpeyev, D., Kaushik, D., Knepley, M.G., May, D.A., McInnes, L.C., Mills, R.T., Munson, T., Rupp, K., Sanan, P., Smith, B.F., Zampini, S., Zhang, H., Zhang, H.: PETSc users manual. Argonne National Laboratory, Tech. Rep. ANL-95/11 - Revision 3.12, (2019). <https://www.mcs.anl.gov/petsc>
28. Birchfield, A.B., Xu, T., Gegner, K.M., Shetye, K.S., Overbye, T.J.: Grid structural characteristics as validation criteria for synthetic networks. IEEE Trans. Power Syst. 32(4), 3258–3265 (2017)
29. Xu, T., Birchfield, A.B., Gegner, K.M., Shetye, K.S., Overbye, T.J.: Application of large-scale synthetic power system models for energy economic studies. Paper presented at the 50th Hawaii International Conference on System Sciences (HICSS), Koloa, HI, 21 April 2017
30. Abhyankar, S., Smith, B., Constantinescu, E.: Evaluation of overlapping restricted additive schwarz preconditioning for parallel solution of very large power flow problems. Proceedings of the 3rd International Workshop on High Performance Computing, Networking and Analytics for the Power Grid - HiPCNA-PG 13, (2013)
31. ALCF: Theta supercomputer. <https://www.alcf.anl.gov/support-center/theta-and-thetagpu>, (2021)
32. Sun, X.-H., Zhu, J.: Performance considerations of shared virtual memory machines. IEEE Trans. Parallel Distrib. Syst. 9, 1185–1194 (1995)

How to cite this article: Sundermann, R., Abhyankar, S.G., Zhang, H., Kimn, J.-H., Hansen, T.M.: Parallel primal-dual interior point method for the solution of dynamic optimal power flow. IET Gener. Transm. Distrib. 1–10 (2022).
<https://doi.org/10.1049/gtd2.12708>

APPENDIX A: PDIPM IN PETSc

Portable, extensible toolkit for scientific computation (PETSc) is an open-source library (BSD-style license) for the numerical solution of large-scale applications and provides the building blocks for the implementation of application codes on parallel (and serial) computers. The PETSc package consists of a set of libraries for creating parallel vectors, matrices, distributed arrays, scalable linear, nonlinear and timestepping solvers.

Our work in this paper added PDIPM, a new nonlinear optimization solver, to the PETSc library to assist both future power system applications and general usage. Figure A.1 illustrates the software structure of the PDIPM in PETSc. Within this solver, we iteratively solve the KKT systems (Equation (18)) using a preconditioned Krylov subspace method. The PDIPM solver in PETSc expects the user to provide the following based on the decision variable \mathbf{x} :

- (i) the vectors \mathbf{x}_l and \mathbf{x}_u , lower and upper bounds for each \mathbf{x} ;
- (ii) the function that calculates $f(\mathbf{x})$ and $\nabla f(\mathbf{x})$;
- (iii) the equality $\mathbf{g}(\mathbf{x})$ and inequality $\mathbf{h}(\mathbf{x})$ constraints as separate vectors and their Jacobian matrices; and
- (iv) the Hessian matrix $\nabla^2 f(\mathbf{x})$, the Hessian of the equality constraints and inequality constraints multiplied by their Lagrangian multipliers (\mathbf{W}_{xxx} in Equation (22)).

From here, PETSc assembles the data structures described in Equations (16) – (22) to implement PDIPM. In each iteration of Newton's method, PDIPM updates only the elements of $\mathbf{G}(\mathbf{x})$, $\mathbf{H}(\mathbf{x})$, $\nabla \mathbf{G}(\mathbf{x})$, $\nabla \mathbf{H}(\mathbf{x})$, and \mathbf{W}_{xxx} that directly depend on \mathbf{X}_n .

The parallel implementation of the PDIPM method is achieved by distributing all involved vectors and matrices across

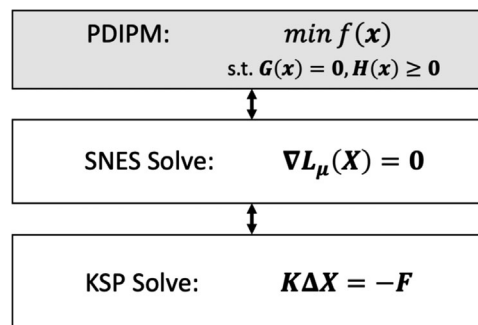


FIGURE A.1 Software structure of PDIPM in PETSc. KSP: Krylov subspace and preconditioner methods. SNES: scalable nonlinear equation solvers

multiple processors and utilizing PETSc scalable linear equation solvers (KSP) and scalable nonlinear equation solvers (SNES) [27].

APPENDIX B: PRECONDITIONER OPTIONS

Within PETSc are numerous options to allow users to select solvers and modify parameters at runtime. The options we used in this paper to apply either the \mathbf{LDL}^T or block-Jacobi preconditioners to the GMRES Krylov subspace iterations are shown below.

B.1 | MUMPS' \mathbf{LDL}^T

```
-ksp_type gmres -pc_type cholesky  
-pc_factor_mat_solver_type mumps
```

B.2 | Block-Jacobi with \mathbf{LDL}^T in the inner subblocks

```
-ksp_type gmres -pc_type bjacobi  
-sub_pc_type cholesky  
-sub_pc_factor_mat_solver_type mumps
```