Scalable PETSc Implementation of the Three-Phase Unbalanced AC Power Flow Solver

[a]Computational Science Center, National Renewable Energy Laboratory
[b]Strategic Energy Analysis Center, National Renewable Energy Laboratory

Justin Chang[a], Gord Stephen[b], Dheepak Krishnamurthy[b], Dan Thom[b], and Wesley Jones[a]

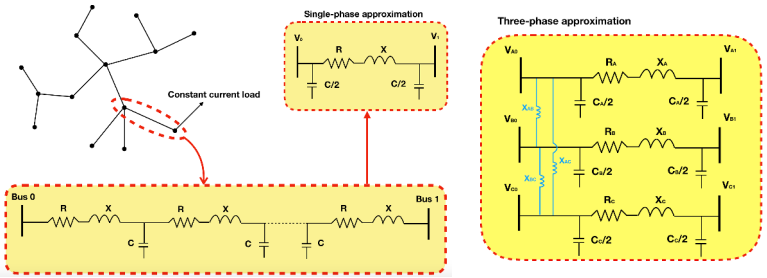
SIAM Conference on Computational Science and Engineering
February 25-March 1, 2019
Spokane Convention Center
Spokane, Washington, USA

Poster # 16

Introduction and Motivation

The growing adoption of behind-the-meter and distributed energy technologies (rooftop solar, battery storage, electric vehicles, etc) is driving a transition in electrical power systems from traditional centralized generation structures to a much more decentralized model of electricity production and consumption. This shift is redefining the role of low-voltage, unbalanced electrical "distribution" networks away from the unidirectional delivery of power and toward a platform that must be capable of accommodating time-varying injections and withdrawals of local power. Fully modelling the impacts of distributed energy resource adoption on both local distribution systems as well as the broader bulk transmission system requires larger and more frequent unbalanced AC power flow calculations, increasing computational requirements. This poster presents preliminary work on designing a robust and scalable 3-phase unbalanced AC power flow solver capable of leveraging high-performance computing resources. The proposed power flow solver will eventually be built on top of the PETSc library and utilize its state-of-the-art parallel data structures like DMNetwork, which uses graph partitioning tools like ParMETIS, that easily interface with several existing iterative solvers and preconditioners.

Schematic of a Power Grid



Symbols:

R = Resistance
X = Reactance
C = Capacitance
I_i= Current at bus i
Y_{ij} = Admittance Matrix
- primitive if i == j
- mutual otherwise
S_i = Power at bus i
V_i = Bus voltage at bus i
n = Total no. of buses

Power Flow Equations

$$\Delta S_i = S_i - S_{i,sched} = S_i - (S_{i,gen} - S_{i,dem})$$

$$S_i = V_i \cdot I_i = \sum_{j=1}^{n} V_i \cdot Y_{ij} \cdot V_j$$

Kirchhoff's Current Law:

$$I_i = Y_{ij} \cdot V_j = \begin{pmatrix} I_1 \\ I_2 \\ \vdots \\ I_n \end{pmatrix} = \begin{bmatrix} Y_{11} & \cdots & Y_{1n} \\ \vdots & \ddots & \vdots \\ Y_{n1} & \cdots & Y_{nn} \end{bmatrix} \begin{pmatrix} V_1 \\ V_2 \\ \vdots \\ V_n \end{pmatrix}$$

Example Y matrix for 3 connected single-phase buses:

$$Y = \begin{bmatrix} y_{11}+y_{12}+y_{13} & -y_{12} & -y_{13} \\ -y_{21} & y_{22}+y_{21}+y_{23} & -y_{23} \\ -y_{31} & -y_{32} & y_{33}+y_{31}+y_{32} \end{bmatrix}$$

Example Y matrix for a 3-phase line:

$$Y_{ij}^{abc} = \begin{bmatrix} Y^{aa} & Y^{ab} & Y^{ac} \\ Y^{ba} & Y^{bb} & Y^{bc} \\ Y^{ca} & Y^{cb} & Y^{cc} \end{bmatrix}_{ij}$$

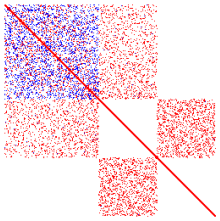
Outline of this work

Our end goal is to develop a robust and scalable distribution system solver, and we wanted to investigate whether PETSc is a viable route to take. To answer this question we took the following preliminary steps:

1. Build an unbalanced distribution system through OpenDSS
2. Extract Y and I. Solve for V using *direct* methods
3. Compared the performance of PETSc against other commonly used *direct* methods

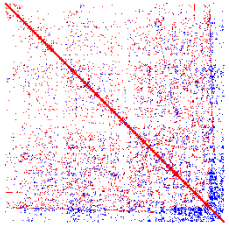
Example admittance matrices

Distribution system 1



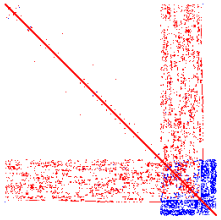
8541 degrees-of-freedom

Distribution system 2



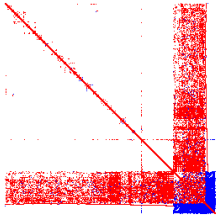
22346 degrees-of-freedom

Distribution system 3



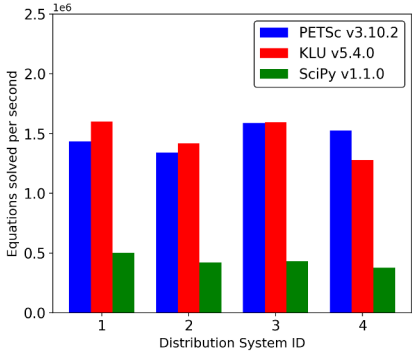
37093 degrees-of-freedom

Distribution system 4



320745 degrees-of-freedom

Comparison of Direct Solvers



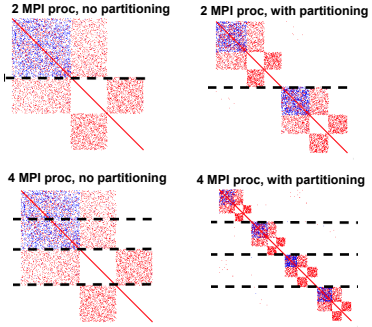
Static-scaling plot comparing the rate at which three different direct solvers process the degrees-of-freedom

Notable observations:

1. SciPy uses UMFPACK as its direct solver
2. KLU is a popular direct solver for circuit and power flow simulations
3. PETSc's native LU solver is slower than KLU for small problems, but appears to algorithmically scale better with problem size

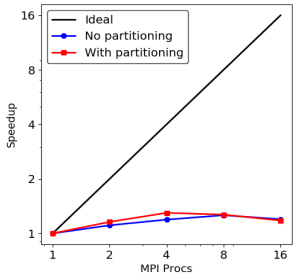
Domain Decomposition

In order to efficiently invert the Y-bus matrices we need a smart way of splitting the matrix across all the MPI processes. Described below are two ways we decompose the Y bus matrices



Partitioned matrices of Distribution System 1 across two and four MPI processes. The LHS column does not use any partitioning techniques whereas the RHS column uses ParMETIS for load balancing.

Parallel speedup using MUMPS



Speedup on a single Haswell node for Distribution System ID 4.

Notable observations:

- Achieved little speedup
- Domain decomposition improved the parallel performance of the direct solver
- Problem still too small to need parallelism
- Need better domain decomposition techniques
- Desirable to have iterative solvers and preconditioners

Current and Future Work

- Based on the results presented in this paper, PETSc appears to be a viable route into developing an open-source scalable distribution system solver.
- Currently developing a framework based on PETSc DMNETWORK
  - Efficient domain decomposition and matrix ordering
  - Leverages the state-of-the-art solvers
- Finding the right iterative solver/preconditioner combo remains an unsolved problem.
- If you have any suggestions on how to solve these types of equations efficiently in parallel, we'd love to hear from you!

E-mail: justin.chang@nrel.gov

This work was authored by the National Renewable Energy Laboratory (NREL), operated by Alliance for Sustainable Energy, LLC, for the U.S. Department of Energy (DOE) under Contract No. DE-AC36-08GO28308. This work was supported by the Laboratory Directed Research and Development (LDRD) Program at NREL. The views expressed in the article do not necessarily represent the views of the DOE or the U.S. Government. The U.S. Government retains and the publisher, by accepting the article for publication, acknowledges that the U.S. Government retains a nonexclusive, paid-up, irrevocable, worldwide license to publish or reproduce the published form of this work, or allow others to do so, for U.S. Government purposes.

SIAM Conference on Computational Science and Engineering
Spokane, Washington
February 25 – March 1, 2019
NREL/PO-2C00-73358