SAND2019-6851C

Implementation of Temporal Parallelization for Rapid Quasi-Static Time-Series (QSTS) Simulations

Joseph A. Azzolini[1], Matthew J. Reno[1], Davis Montenegro[2]

[1] Sandia National Laboratories, Albuquerque, NM, 87185, USA
[2] Electric Power Research Institute, Knoxville, TN, 37932, USA

***Abstract* — Quasi-static time-series (QSTS) analysis of distribution systems can provide critical information about the potential impacts of high penetrations of distributed and renewable resources, like solar photovoltaic systems. However, running high-resolution yearlong QSTS simulations of large distribution feeders can be prohibitively burdensome due to long computation times. Temporal parallelization of QSTS simulations is one possible solution to overcome this obstacle. QSTS simulations can be divided into multiple sections, e.g. into four equal parts of the year, and solved simultaneously with parallel computing. The challenge is that each time the simulation is divided, error is introduced. This paper presents various initialization methods for reducing the error associated with temporal parallelization of QSTS simulations and characterizes performance across multiple distribution circuits and several different computers with varying architectures.**

***Index Terms* — distribution system modeling, quasi-static time-series, PV grid integration, parallel processing.**

I. Introduction

Quasi-static time-series (QSTS) simulation is a powerful study tool for modern distribution system analysis. A QSTS simulation solves a series of sequential steady-state power-flow solutions where the converged state of each iteration serves as the initial state of the following iteration [1]. QSTS simulations are particularly useful when analyzing circuits with new smart grid technologies and/or high penetrations of distributed and renewable resources due to their ability to model circuit elements with discrete controls. Since the simulation is run as a time-series, the time-varying parameters like load and photovoltaic (PV) generation, as well as the time-dependent states of circuit elements, like regulator tap positions, can be captured for analysis. Time-series analysis of distribution circuits is becoming increasingly important as more distributed energy resources (DER) like residential rooftop PV, energy storage, and smart inverters are connected to the grid.

QSTS simulations have many benefits but can be prohibitively burdensome for large and complex circuits due to long computation times. To capture all distribution system analyses accurately, a QSTS simulation with a time horizon of one year and a time step resolution of 5 seconds or less is recommended [2]. This simulation requires millions of power-flow solutions to be solved, incurring long computation times. To address this challenge, a number of different methods have been explored in the literature: circuit reduction can be implemented to reduce the number of buses [3], machine learning can be used in partial-year simulations to predict the results for full yearlong simulations [4], the circuit can be divided into smaller sections and solved concurrently across multiple processors using Diakoptics [5], and the number of power flows can be reduced using a number of different algorithms [6]–[9].

Temporal parallelization of QSTS simulations is another promising method for reducing long computational times by dividing the total time horizon of the simulation into different sections and solving those sections simultaneously on multiple processors of a single computer. Fig. 1 shows a visual representation of a QSTS simulation divided into four equal parts to be solved concurrently on four separate processors. In this case, each section of the simulation is performed independently and in parallel to one another.

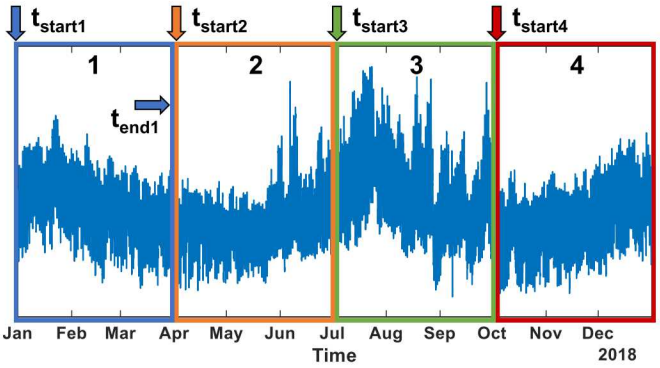

Fig. 1. Yearly QSTS simulation with four temporal divisions.

Dividing a single simulation into multiple sections introduces discontinuities into the simulation—specifically at the beginning of each temporal division. The initial conditions at these temporal divisions are not known, thus error is introduced. For example, in Fig. 1, the initial state at "t_{start2}", "t_{start3}", and "t_{start4}" are unknown. Since each power-flow solution may have more than one valid solution (especially when the circuit has multiple controllable devices), the calculated states may differ from the actual base-case states. These differences can cause error to persist through the simulation because each converged state serves as the initial state for the following power flow solution. As more temporal divisions are introduced, more unknown states are introduced and the likelihood of error increases.

Finding the correct initial state for each divided section is critical for reducing these errors. In [10], a Monte-Carlo based approach was used to set the initial state of each section and reduce the overall parallelization error. Another method to

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

reduce parallelization error is by running a short-term QSTS simulation to initialize each section, as shown in Fig. 2. In this case, the difference between "t_{start2}" and "t_{end1}" represents the initialization window. In one study, the authors found that longer initialization times (24 hours compared to 90 minutes) had performed better for error reduction [11]. The down side is that longer initialization windows take longer to compute, reducing the benefits of temporal parallelization.

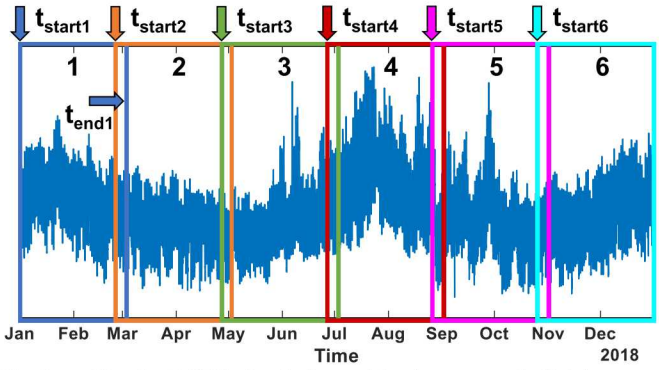

Fig. 2. Yearly QSTS simulation with six temporal divisions and initialization windows.

This work focuses on the implementation of temporal parallelization, across multiple test feeders and computers with different characteristics, to quantify the benefits and limitations of this approach for rapid QSTS simulations. The contributions of this paper include:

- Characterization of error and speed improvement for temporally parallelized QSTS simulations on multiple distribution feeders
- Analysis and verification of speed and accuracy across several computers
- Comparison of initialization strategies for reducing error
- Test circuits and temporal parallelization code are open source and publicly available to download [12]

The paper is organized as follows: Section II provides details of how temporal parallelization is implemented in MATLAB and OpenDSS, Section III describes the experimental methodology of the paper, Section IV presents the results for parallelization error, Section V presents the results for speed improvement, and Section VI concludes the paper.

II. Implementation of Temporal Parallelization

The temporally parallelized QSTS simulations on all computers were set up using the GridPV toolbox [16] in MATLAB R2018b and were run in OpenDSS version 8.5.9.1 (64-bit build) [17]. This version of OpenDSS has built-in capability for running parallel QSTS simulations, based on an actor model [18]. Each actor created by OpenDSS has its own thread and can be individually assigned a CPU to run on.

Fig. 3 shows an overview of how to run a QSTS simulation using the parallel processing functions in OpenDSS. Additional information regarding these functions can be found in [19]. First, a circuit must be compiled that is set up for QSTS simulations, i.e. it contains time-series profiles for any loads or generators in the model. After a circuit is compiled, the *clone* function can be used to copy this model onto additional actors. Each actor is essentially another instance of OpenDSS, running on a separate processor. The actors are centrally controlled via the parallelization interface, where commands can be sent to a specific actor (e.g. by setting *ActiveActor=4*) or to all actors at once by setting *ActiveActor=**. Once the actors have been created, monitors can be added to log the time-series data of various circuit components across all actors. If the monitor's name is the same in every actor, the results can be exported into a single file by setting *ConcatenateReports=true*.

Each actor can be individually initialized and set up for a QSTS simulation (several initialization strategies are discussed in Section IV). It is best practice to set the *ActiveActor=1* before issuing the *solveAll* command to ensure proper execution. While the actors are solving their assigned portions of the QSTS simulation, their progress can be tracked using the *ActorProgress* command. Upon completion of the simulation, the *ActorStatus* of each actor returns a 1, letting the user know that those actors are ready to accept new commands, e.g. exporting the data from the monitors for analysis.

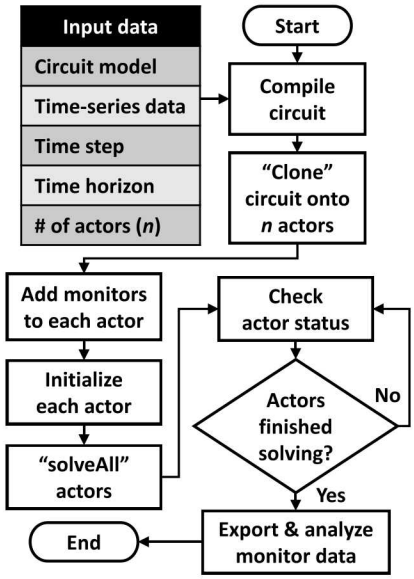

Fig. 3. Flowchart for implementing temporal parallelization for QSTS simulations in OpenDSS.

In general, the performance of parallel computing does not scale linearly with the number of parallel divisions. The performance can be affected by various factors such as the need to share certain resources, e.g. memory, or if the application itself contains a serial component that cannot be divided across multiple processors. One way to capture the upper bounds of expected system performance is to use an abstract analytic method like Amdahl's law [20], shown in (1). This law states that *Times Faster* scales up based on the number of parallel divisions, *N*, and the serial component of the application, *S*.

$$Times\ Faster = \frac{1}{S+\frac{(1-S)}{N}} \quad (1)$$

Fig. 4 gives some examples of Amdahl's law for processes with varying percentages of serial components as the number of parallel divisions being used increases. The dashed blue line represents the theoretical maximum speed improvement of temporal parallelization, i.e. if the QSTS simulation was perfectly parallelizable and no computational resources needed to be shared. This figure shows that parallelization performs best when the percent serial component is as small as possible, and that there is a diminishing marginal improvement with each additional parallel division.

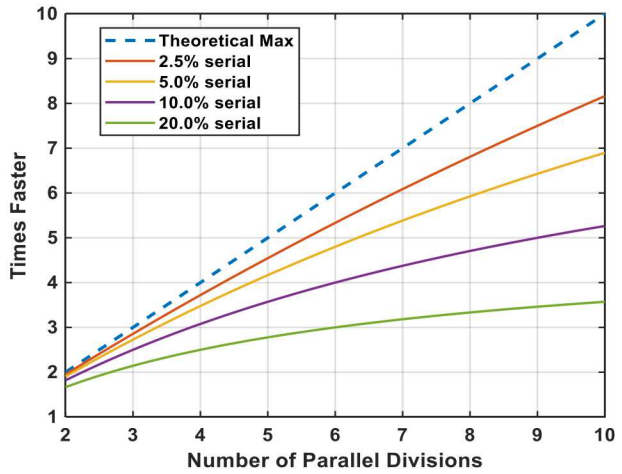

Fig. 4. Expected speed improvements with increasing number of parallel divisions, as predicted by Amdahl's law (1) for several values of *S*.

III. Experimental Methodology

To accurately quantify the performance of temporal parallelization, QSTS simulations were run on four different test circuits. Each of the four test circuits have been individually modified for QSTS simulations, i.e. adding time-series profiles for loads and PV systems, and are publicly available [12]. The simulations for one of the test circuits were then repeated across two other computers with differing characteristics for comparison and verification. The following subsections provide more detail on each of the test circuits, computers used for the simulations, and evaluation criteria.

A. Test Circuits

The first test circuit used, Feeder CO1, is a 21.7 km-long circuit based on an actual utility distribution feeder [13]. This circuit has 2969 buses (5469 nodes), 9 controllable elements (switching capacitor banks and step voltage regulators), 144 PV systems (3.7 MW total), and a peak load of 6 MW. Fig. 5 shows a circuit diagram of the distribution feeder, with various circuit elements highlighted. The black lines represent 3-phase distribution lines and the gray lines represent the 1 or 2-phase secondary system. Each load is assigned a 1-second power injection profile by customer class (residential or commercial) generated from measurements of the feeder under consideration. Each PV system is grouped into one of four categories based on its geographic location and assigned a unique 1-second power injection profile based on solar irradiance data. See [7] for more details on the circuit.

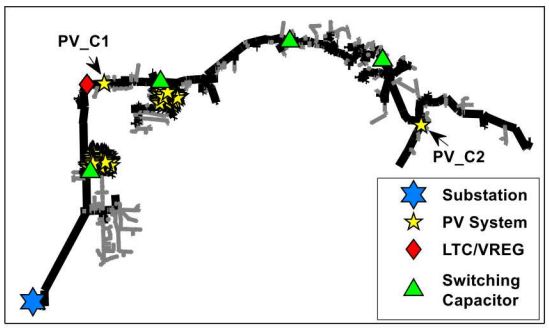

Fig. 5. Circuit plot of Feeder CO1 (and Feeder CO1-VV) marked with locations of PV systems, voltage regulators, and switching capacitors.

The second test circuit, Feeder CO1-VV, is the same as Feeder CO1 except the two centralized PV systems (PV_C1 and PV_C2 in Fig. 5) were modeled with Volt-Var functionality according to IEEE 1547 [1]. These two PV systems are 3-phase, centralized installations with their own interconnection transformers.

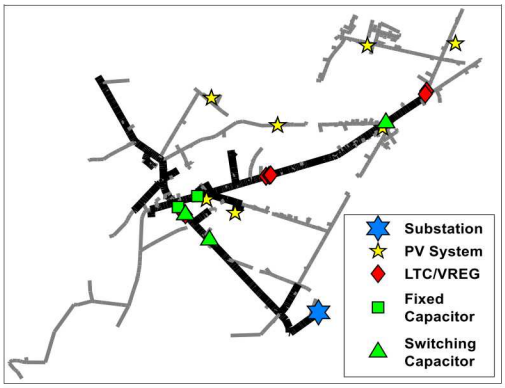

Fig. 6. Circuit plot of the modified EPRI Feeder J1 circuit.

The third circuit, shown in Fig. 6, is a modified version of the EPRI Feeder J1 test circuit [14]. This circuit, also based on actual utility distribution feeder, has 3433 buses, 7 PV systems (1.77 MW total) each with their own unique time-series profile, 12 controllable elements, and a peak load of 6.3 MW.

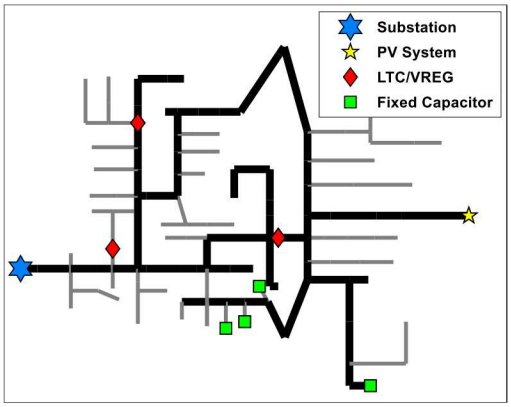

Fig. 7. Circuit plot of modified IEEE123 circuit.

The last of the four test circuits is a modified version of the IEEE 123-bus test feeder [15]. This circuit is unbalanced, with loads on each phase having their own unique time-series profiles. The IEEE 123-bus network, shown in Fig. 7, has a 4.16 kV operating voltage, 7 step voltage regulators, one 1.8 MW PV system with a constant power factor of -0.98, and a peak load of 3.49 MW.

B. Computers Used for Simulations

To fully characterize and verify the speed improvements of temporal parallelization, simulations were run on three different computers. Table I shows a summary of key characteristics of the three computers used for parallelized QSTS simulations. All three computers are also equipped with hyperthreading technology, meaning that each core contains two logical processors (CPUs) and each CPU can support its own actor in OpenDSS. To avoid having the computers freeze up during simulation, all CPUs should not be used at once for an OpenDSS simulation [19]. Thus, the maximum number of actors each computer can support can be found using (2).

$$Maximum\ Actors = (2 * Cores) - 1 \tag{2}$$

It should noted that Computer C has two sockets each with 20 cores for a total of 40 available cores. However, some of the libraries being used are unable to recognize and simultaneously communicate with both sockets. Hence, only 20 cores of Computer C are used for experimentation.

TABLE I. SUMMARY OF COMPUTER CHARACTERISTICS

Name	CPU	Base Speed	Memory	# of Cores
A	Intel® Core™ i7-4790	3.60 GHz	16 GB	4
B	Intel® Xeon® E5-2687W v3	2.10 GHz	32 GB	10
C	Intel® Xeon® Gold 6138	2.00 GHz	96 GB	40*

**Due to some library limitations, only half of the cores are available for experimentation*

C. Evaluation Criteria

Each temporally parallelized QSTS simulation is evaluated on its speed and accuracy by comparing the results to a base-case simulation of the same circuit on the same computer. The base case simulation for each circuit had a time horizon of one year and used a time step resolution of 1-second, for a total of 31,536,000 power flow solutions. By using a 1-second time step resolution (the finest resolution allowed for QSTS simulations [1]), any errors in parallelized simulations can be directly attributed to the parallelization process.

The parallelization errors are demonstrated by monitoring the voltages of two different nodes on the circuit (one for over-voltages and one for under-voltages), the total number of yearly tap position changes of the step voltage regulators, and total number of switching events from the switching capacitors. The monitored nodes were chosen after analyzing the base-case results to see which nodes experienced the highest and lowest voltages throughout the year.

Acceptable error thresholds for the criteria listed above have been set based on feedback from distribution system engineers on their expectations of the performance of QSTS simulations [9]. Since each circuit has multiple step voltage regulators and capacitors, root-mean-square error (RMSE) is used to encapsulate the errors of all devices into a single number. In (3), n represents the number measurement points, such as the number of regulators or capacitors, P represents the measurement for the parallelized simulation and O represents the base-case measurement. For example, when calculating RMSE for tap position change, n would be the number of regulators in the circuit, P_i would be the tap changes of the i^{th} regulator in the parallelized simulation, and O_i would be the base-case tap changes for the i^{th} regulator. A similar equation (4) can be used to show RMSE in terms of percent error.

$$RMSE = \sqrt{\frac{\Sigma_{i=1}^{n}(P_i - O_i)^2}{n}} \tag{3}$$

$$RMSE\ \% = \sqrt{\frac{\Sigma_{i=1}^{n}\left(\left(\frac{P_i - O_i}{O_i}\right)\times 100\right)^2}{n}} \tag{4}$$

To quantify speed improvements, base-case simulations were run on a single core using a time horizon of one year and a time step resolution of 1 second. For parallelized simulations, *Times Faster* was calculated using (5). Even though each actor is solving an equal amount of power flows, it is not guaranteed that they take the same amount of time. Therefore, the slowest actor is used to calculate *Times Faster* in (5).

$$Times\ Faster = \frac{SimulationTime_{base}}{\max(SimulationTime_{actors})} \tag{5}$$

IV. RESULTS FOR PARALLELIZATION ERROR

The following subsections highlight the error results of the temporally parallelized QSTS simulations on multiple test circuits. First, in Subsection A, several initialization strategies were investigated for their ability to reduce parallelization error. Then, one strategy was selected and used for the remainder of the simulations described in Subsection B.

A. Initialization Strategies for Reducing Error

Several initialization strategies were investigated to reduce the error associated with parallelization. The first method involves a single power-flow solution using the "static" control mode before changing to "time" mode for the remainder of the QSTS simulation. In "time" mode, control actions are executed when the time for the pending action is reached or surpassed. This mode reflects how controllable devices behave in actual distribution systems. In "static" mode, control actions are executed in the order of shortest time to act until all control actions are cleared from the control queue [17]. This mode is

useful in circuits with multiple controllable devices like voltage regulators and switching capacitors as it allows these devices to settle to a converged state without advancing time before the rest of the simulation begins. The initialization time of this method is negligible since it only requires a single power flow solution.

The next initialization method utilizes the static solution and a one-day long initialization window. First, the actor rewinds to the day before it is tasked to solve. For example, if the actor was assigned to start on Day 100, it first rewinds to Day 99, solve one static power-flow, then solve the remainder of Day 99 at a 5-second time step resolution in "time" mode until it reaches Day 100. The 5-second time step resolution was chosen to speed up the initialization while remaining less than the shortest delay time of any controllable element (30 seconds in this case). This initialization method is more computationally intensive than the first, as it requires 17,280 initialization power-flow solutions.

The final initialization method is conceptually the same as the previous one, except that it rewinds one week instead of one day and uses a 30-second time step resolution. This method is the most computationally intensive of the proposed methods, as it requires 20,160 initialization power-flow solutions.

The results of each of the proposed methods are compared to each other and the case without any initialization in Fig. 8. This figure shows that the voltage regulator error and switching capacitor error both increase with the number of parallel actors being used, as expected. However, even in the worst case considered here, the error magnitudes are still well within the acceptable thresholds (10% for regulator RMSE and 20% for capacitor RMSE [9]). Thus, the "static" initialization method was chosen for all subsequent simulations since it reduces the error without adding any computational time.

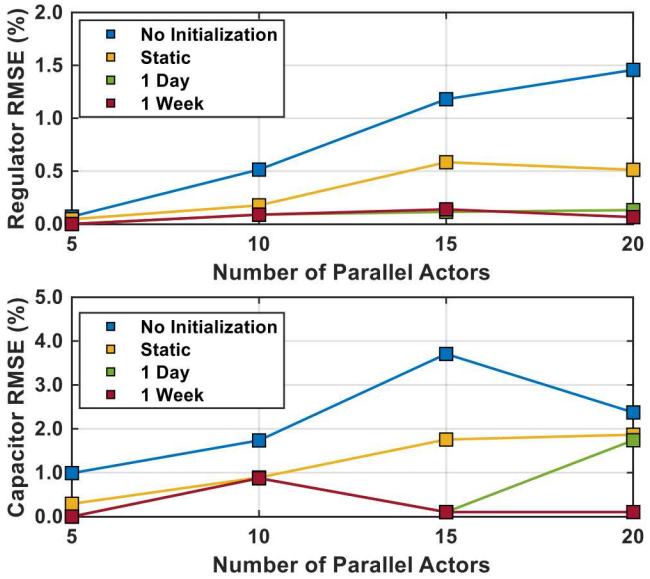

Fig. 8. Feeder CO1 RMSE of voltage regulator tap position changes (top) and RMSE of capacitor switches (bottom) of the proposed initialization strategies with increasing number of parallel actors on Computer C.

B. *Parallelization Error of Each Test Circuit*

As opposed to snapshot analyses, QSTS simulations can capture the time-dependent states of the systems that can be used to study important characteristics of modern distribution circuits like the effects of daily changes in load and PV output and the duration of extreme conditions on the circuit. One key metric provided by QSTS simulations is the number of tap position changes throughout the year of step voltage regulators. These devices adjust their tap position to maintain voltages within certain predetermined limits. As variability increases on a circuit due to high penetrations of DER, voltage regulators tend to operate more frequently which can affect their lifetime. Thus, the ability to accurately characterize their operation throughout the year is critical. Fig. 9 shows the regulator tap position changes RMSE for each of the test circuits with increasing levels of parallelization. While the errors do tend to increase with the number of actors, overall the error is well within the acceptable limits (represented by the horizontal dashed black line).

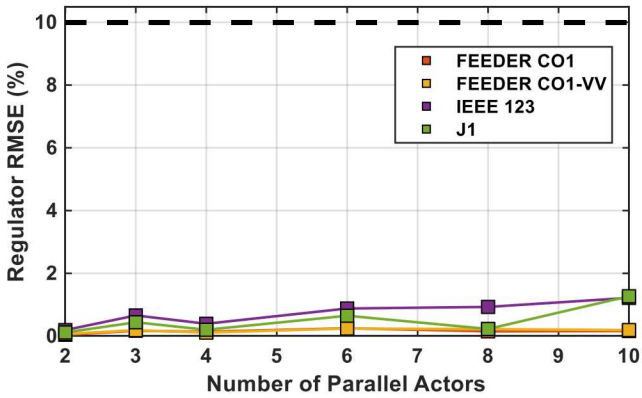

Fig. 9. Regulator tap position change RMSE with increasing number of actors (Computer B).

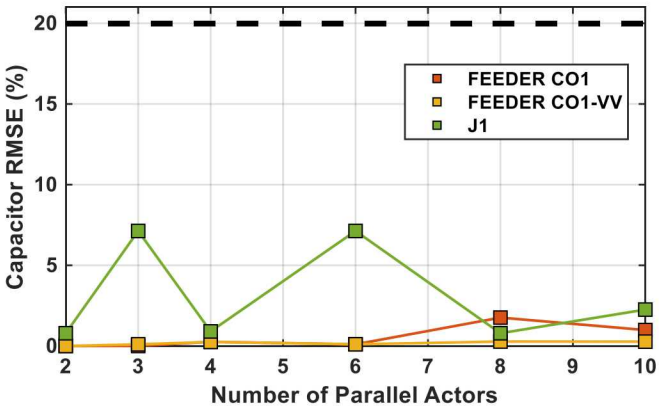

Fig. 10. Switching capacitor state change RMSE with increasing number of actors (Computer B).

Similar to the step voltage regulators, switching capacitors are also used help maintain voltages along a feeder and their number of operations can increase with higher variability on the circuit as well. Fig. 10 shows the switching capacitor state change RMSE of three different test circuits with increasing

levels of parallelization (the IEEE 123 circuit does not contain any switching capacitors). Again, the errors are all within the 20% limit. It should be noted that the switching capacitors in the J1 test circuit have fewer operations than those in the other test circuits, causing this metric to be more sensitive to the *RMSE %* calculation in (4).

Voltage violations are defined by the amount of time the voltage on a distribution circuit spends outside its predetermined limits, such as those set forth in ANSI C84.1. To capture this behavior, monitors were added to two different nodes in the circuit to capture over-voltages and under-voltages. The error metric for voltage violations (measured in hours) was calculated by taking the difference between the parallelized simulations and the base-case for each of the test circuits. Fig. 11 shows the voltage violation error results for each of the test circuits with increasing levels of parallelization. While the errors are all within the limit of 24 hours, two of the test circuits experienced a sharp increase from 8 to 10 actors. Further investigation would be needed to tell if these errors would continue to grow with additional actors.

Along with the duration of extreme voltages, the magnitude of those voltages was also considered. The RMSE for the yearly maximum and minimum voltages on the feeder were each accurately captured by all the parallelized simulations, as shown in Fig. 12.

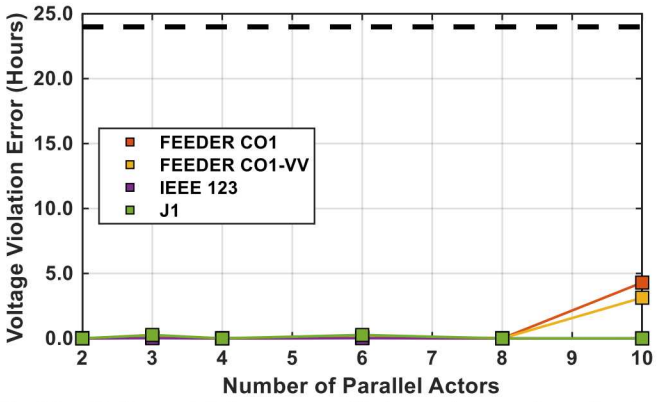

Fig. 11. Voltage violation error with increasing number of actors (Computer B).

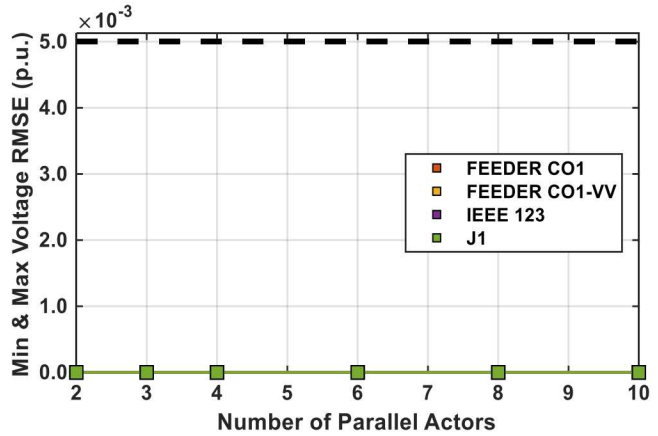

Fig. 12. Minimum and maximum voltage RMSE with increasing number of actors (Computer B).

V. Results for Speed Improvement

There are a variety of factors that influence the speed improvements of parallelized QSTS simulations including the size of the circuit, computing resources available, thread allocation, and actor solution times. These factors will be discussed in detail in the following subsections.

A. Actor Solution Times

Even though each actor in a parallelized QSTS simulation is solving the same number of power flows, the solution time per power flow is not constant. Each power flow may require a varying number of iterations to converge on a solution depending on circuit conditions or may need to account for actions taken by controllable devices. An example of this behavior is shown in Fig. 13 where Actor 3 finished more than 20 minutes after Actor 10. Thus, the computation time of the slowest actor is used to calculate *Times Faster* in (5). Overall, this phenomenon further reduces the performance predictions from Amdahl's law.

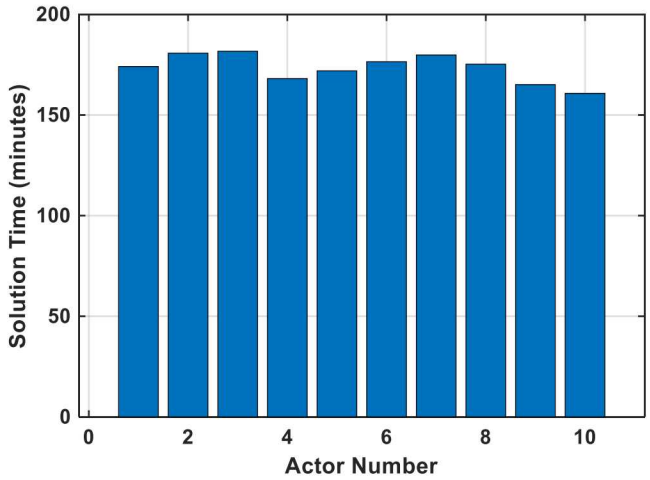

Fig. 13. Comparison of actor solution times for Feeder CO1 on Computer B with 10 actors.

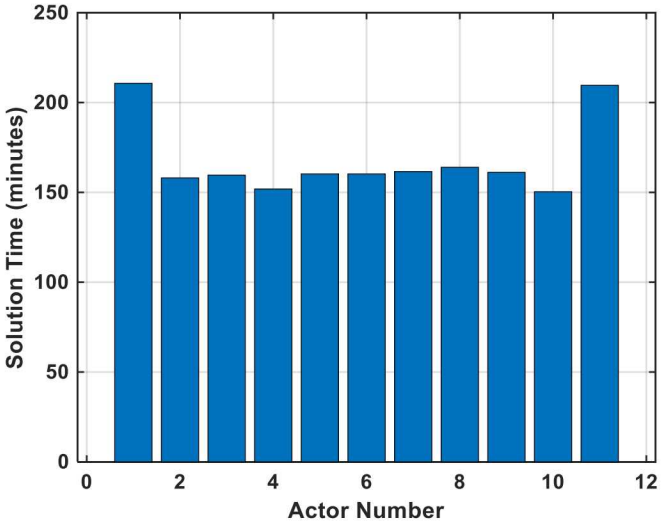

Fig. 14. Comparison of actor solution times for Feeder CO1 on Computer B with 11 actors.

Another factor that can impact actor solution time is thread allocation. Since every actor is running on its own thread, the ideal situation is to have each thread running on its own core, as in Fig. 13. Since Computer B has 10 cores and the simulation had 10 actors, each actor in was assigned to a different core. According to (2), Computer B can support up to 19 actors. So, for any parallelized simulations on Computer B with more than 10 actors, at least one of the cores will be supporting two threads instead of one.

Fig. 14 shows the solution times of each of the 11 actors. This figure shows that Actor 1 and Actor 11 were assigned to run on the same core. As expected, these two actors had the longest solution times; Actor 1 finished more than an hour after the Actor 10 and the total speed improvement was *worse* with 11 actors than it was for 10 actors.

To further quantify the effects of thread allocation, a batch of simulations were run in which no actor had its own core. The results from these simulations are shown in Fig. 15. The results of each case were also individually fit to Amdahl's law (1). The fit coefficient for each case, i.e. the percent serial component, are shown in the legend. Fig. 15 reveals that the benefits of having additional actors eventually outweigh the cost of the increased serial component when using two threads per core.

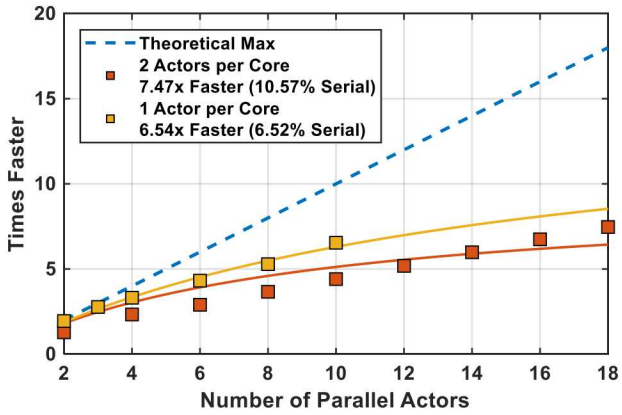

Fig. 15. Effects of thread allocation on speed improvement using Computer B, with results fit to Amdahl's law (1).

B. Circuit Results

The speed improvements of each test circuit under various levels of parallelization are shown in Fig. 16. The results of each circuit were individually fit to Amdahl's law (1). The fit coefficient for each circuit and the R^2 value for the fit are summarized in Table II. These results show that speed improvements were not consistent across the various test circuits and that parallelization was more beneficial for the larger circuits. In general, the power flow solution time tends to increase with the number of buses in the circuit model, i.e. a single power flow solution takes longer for Feeder CO1 than it does for the IEEE 123 circuit. Therefore, the serial component of QSTS simulations, such as loading the time-series profile values from memory, plays a more dominant role in smaller circuits. This behavior is evident in Table II as the IEEE 123 test circuit had a 9.40% serial component compared to the 6.52% serial component in Feeder CO1.

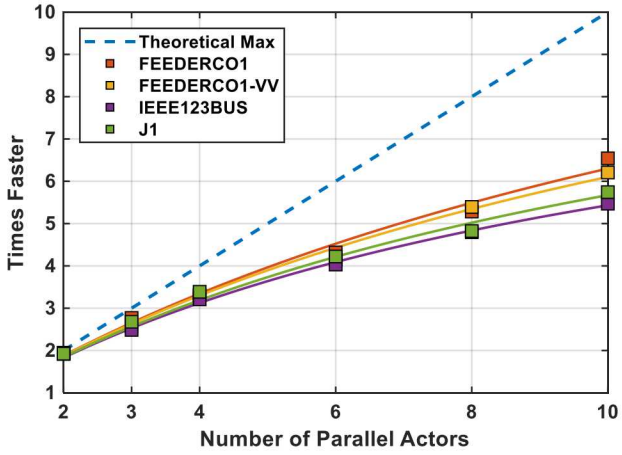

Fig. 16. Speed improvements of all circuits at various levels of parallelization simulated on Computer B with each fit to (1).

TABLE II. DETAILED RESULTS FROM FIG. 16

Test Circuit	Times Faster (10 cores)	Amdahl Coefficient	R^2 Value
Feeder CO1	6.542x	6.52% serial	0.9887
Feeder CO1-VV	6.208x	7.09% serial	0.9928
Feeder J1	5.744x	8.48% serial	0.9896
IEEE 123	5.370x	9.40% serial	0.9974

C. Computer Assessment

Feeder CO1 was chosen as the test circuit to be used across the all three computers. The results from these simulations are shown in Fig. 17 and Table III. Since the results from Computer B and Computer C are very similar, it is probable that Computer A had other hardware limitations, like memory or cache size, that affected its performance.

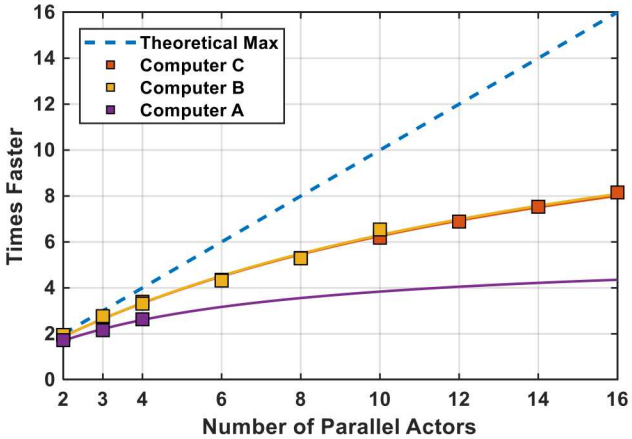

Fig. 17. Speed improvements of Feeder CO1 at various levels of parallelization simulated on all three computers, fit to (1).

TABLE III. DETAILED RESULTS FROM FIG. 17

Computer	Times Faster	Amdahl Coefficient	R^2 Value
A	2.633x (4 cores)	17.84% serial	0.9894
B	6.542x (10 cores)	6.52% serial	0.9887
C	8.156x (16 cores)	6.65% serial	0.9975

VI. Conclusion

QSTS simulations are an important study tool for distribution system analysis, but they can be burdensome with long computation times. This paper explored the method of temporal parallelization as a possible solution to speed up the simulations. However, since QSTS simulations require millions of power flow solutions to be computed sequentially due to their time-dependent nature, parallelizing this process can introduce errors that persist through the simulation.

First, several initialization strategies with varying computational requirements were explored to reduce this error. It was found that solving each divided section using a "static" control mode for one single power flow was able to reduce the parallelization error without adding any significant computation time. This initialization strategy was used for all the remaining parallelized QSTS simulations.

Four different test circuits and three different computers were then used to quantify the speed improvements and accuracy of temporal parallelization. These test circuits and the MATLAB code used to implement temporal parallelization are publicly available [12]. The results from each test circuit were fit to Amdahl's law to characterize their speed improvements. As expected, the larger circuits experienced a greater benefit from parallelization. Temporally parallelized QSTS simulations were shown to have a serial component of at least 6.52% resulting in a speed improvement of up to 8.156x faster when using 16 cores. The parallelized results were also subject to several error metrics. In every case, the parallelization error was found to be within acceptable limits. Based on these results, temporal parallelization has shown to be a viable option for rapid QSTS simulations.

References

[1] IEEE P1547.7/D11, "IEEE Guide for Conducting Distribution Impact Studies for Distributed Resource Interconnection," 2013.

[2] M. J. Reno, J. Deboever, and B. Mather, "Motivation and requirements for quasi-static time series (QSTS) for distribution system analysis," in *IEEE Power and Energy Society General Meeting*, 2017, pp. 1–5.

[3] Z. K. Pecenak, V. R. Disfani, M. J. Reno, and J. Kleissl, "Multiphase Distribution Feeder Reduction," *IEEE Trans. Power Syst.*, vol. 33, no. 2, pp. 1320–1328, 2018.

[4] L. Blakely, M. J. Reno, and R. J. Broderick, "Decision Tree Ensemble Machine Learning for Rapid QSTS Simulations," in *2018 IEEE Power & Energy Society Innovative Smart Grid Technologies Conference (ISGT)*, 2018.

[5] D. Montenegro, G. A. Ramos, and S. Bacha, "A-Diakoptics for the Multicore Sequential-Time Simulation of Microgrids Within Large Distribution Systems," *IEEE Trans. Smart Grid*, vol. 8, no. 3, pp. 1211–1219, 2017.

[6] D. Montenegro, J. Gonzalez, and R. Dugan, "Multi-rate control mode for maintaining fidelity in Quasi-Static-Time-Simulations," *2017 3rd IEEE Work. Power Electron. Power Qual. Appl. PEPQA 2017 - Proc.*, 2017.

[7] M. U. Qureshi, S. Grijalva, M. J. Reno, J. Deboever, X. Zhang, and R. Broderick, "A Fast Scalable Quasi-Static Time Series Analysis Method for PV Impact Studies using Linear Sensitivity Model," *IEEE Trans. Sustain. Energy*, vol. 10, no. 1, pp. 301–310, 2018.

[8] J. Deboever, S. Grijalva, M. J. Reno, and R. J. Broderick, "Fast Quasi-Static Time-Series (QSTS) for yearlong PV impact studies using vector quantization," *Sol. Energy*, vol. 159, no. June 2017, pp. 538–547, 2018.

[9] M. J. Reno, J. A. Azzolini, and B. Mather, "Variable Time-Step Implementation for Rapid Quasi-Static Time-Series (QSTS) Simulations of Distributed PV," in *2018 IEEE 7th World Conference on Photovoltaic Energy Conversion (WCPEC)*, 2018, pp. 1626–1631.

[10] A. Latif and B. Mather, "Monte carlo based method for parallelizing quasi-static time-series power system simulations," *2018 Int. Conf. Probabilistic Methods Appl. to Power Syst. PMAPS 2018 - Proc.*, pp. 1–6, 2018.

[11] R. Hunsberger and B. Mather, "Temporal Decomposition of a Distribution System Quasi-Static Time-Series Simulation," in *2017 IEEE Power & Energy Society General Meeting*, 2017, pp. 5–9.

[12] "Quasi-Static Time-Series (QSTS)," *PVPerformance Modeling Collaborative*, 2019. [Online]. Available: https://pvpmc.sandia.gov/pv-research/quasi-static-time-series-qsts/.

[13] M. J. Reno, K. Coogan, J. Seuss, and R. J. Broderick, "Novel Methods to Determine Feeder Locational PV Hosting Capacity and PV Impact Signatures," *Sandia National Laboratories*, vol. SAND2017-4, 2016.

[14] "Distributed PV Monitoring and Feeder Analysis," *Electric Power Research Institute*, 2019. [Online]. Available: https://dpv.epri.com/feeder_j.html. [Accessed: 06-Mar-2019].

[15] K. P. Schneider *et al.*, "Analytic Considerations and Design Basis for the IEEE Distribution Test Feeders," *IEEE Trans. Power Syst.*, vol. 33, no. 3, pp. 3181–3188, 2018.

[16] M. J. Reno and K. Coogan, "Grid Integrated Distributed PV (GridPV)," *Sandia National Laboratories*, vol. SAND20 13, no. August, pp. 1–134, 2013.

[17] R. C. Dugan, "The Open Distribution System Simulator (OpenDSS)," *Electr. Power Res. Institute, Inc.*, no. November, pp. 1–177, 2012.

[18] D. Montenegro, R. C. Dugan, and M. J. Reno, "Open Source Tools for High Performance Quasi-Static Time-Series Simulation Using Parallel Processing," *IEEE Photovolt. Spec. Conf.*, pp. 3055–3060, 2017.

[19] D. Montenegro and R. C. Dugan, "User Instructions for Parallel Processing," 2018. [Online]. Available: https://sourceforge.net/projects/electricdss/files/. [Accessed: 06-Dec-2019].

[20] G. M. Amdahl, "Validity of the single processor approach to achieving large scale computing capabilities," in *Proceedings of Am. Federation of Information Processing Societies Conf*, 1967, pp. 483–485.

This research was supported by the DOE SunShot Initiative, under agreement 30691. Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.