Abstract—Transient simulation of power systems can be challenging. As models become more complex, applications require acceleration or faster than real time operations to assist system operators, and new methods of solving the problem are required. One potential approach includes time-parallel multigrid reduction methods that aim to distribute the simulation time domain onto multiple compute units and solve the system equations with a parallel, iterative nonlinear multigrid scheme. Achieving significant speedup via these methods for real world power system problems requires examination of how the different features and characteristics of power system simulations interact with the parallel-in-time calculations and how they can be handled by such tools. One such characteristic is the use of unscheduled events that change the system at solution-dependent times. Examples of such events include equipment limits, governor deadbands, or protection system actions. In sequential simulations, such events are detected by using root-finding methods. In time-parallel methods these roots must be handled in parallel. In this paper we propose a method for addressing unscheduled events in time-parallel calculations for power system transient simulation.

I. INTRODUCTION

Transient power grid simulation is a key tool for assessing the dynamic behavior of the power grid. It is used for exploring the impact of faults, ensuring system stability during component failure, and general understanding of the dynamic behavior of power under different conditions. In many situations, accelerating the simulation of power system dynamics can lead to their application in new domains. For example, if transient simulation could be reliably made to work faster than real time for large scale complex models, the results could be used for predictive control or testing control actions. Faster time-to-solution potentially allows for more accurate solutions, longer simulation time executions, or application of simulations in new domains, such as real time controls or dynamic state estimation.

One avenue of research aims to utilize high-performance computing (HPC) to accelerate simulation runtimes by parallelization across the time domain of the underlying transient model. Early work focusing on parallel-in-time integration for transient power grid simulation has been reported in [1], [2], where groups of time-points are solved simultaneously on multiple compute units and a nested multilevel approach is used to solve a cascade of finer and finer time-grid simulations. More recently, the parareal method [3] has been applied to speed up power grid simulations in [4], with further development in [5] proposing to use simpler reduced models on coarser grid levels.

In this paper, we focus on multigrid reduction in time (MGRIT, [6]) which applies nonlinear multigrid iterations to the time domain of the discretized power system equations. In its two-level variant, MGRIT is equivalent to parareal - the multilevel distinction, however, is important to achieve a highly scalable method with optimal parallel communication behavior, as opposed to a two-level scheme, where the size of the coarse level limits concurrency.

Previous work on MGRIT with transient power grid systems involved parallel-in-time simulation with scheduled events, such as sudden load changes happening at prescribed time points. Results [7], [8], [9] show fast convergence of the multigrid iterations, independent of the time-domain length and number of scheduled discontinuities. Speedups of the time-parallel code compared to the time-serial version have been reported up to a factor of 50 [7].

In this paper, we develop an approach for incorporating unscheduled events into the multigrid reduction scheme. Here, the locations of the events are not known ahead of time and must also be solved for as part of the solution strategy. In the sequential setting, this is done one event at a time using a root-finding technique. Our approach integrates this root-finding method into the parallel MGRIT algorithm and combines it with a time-adaptive approach that attempts to minimize the number of time points needed while retaining simulation accuracy and uniform multigrid convergence behavior. The result is a method that simultaneously finds the locations of the events while solving the power grid problem in parallel to speed up the overall simulation.

II. PROBLEM DESCRIPTION

Power systems are typically modeled as a system of differential algebraic equations (DAEs) of the form

\[ \dot{x} = f(x, z) \]  

\[ 0 = g(x, z) \]  

which include transient ordinary differential equations (ODEs) describing, for example, synchronous machines, excitation systems, dynamic loads etc., and a set of algebraic equations,
such as the power balance equations describing the network transmission. Typically, power system models are subject to equipment limitations in the sense that some components of $x$ are bounded to be in a certain regime:

$$x_i \in [x_i^{\text{min}}, x_i^{\text{max}}]$$

for some components $i$. Examples of these models include exciter limits where the maximum and minimum excitation voltage or currents are capped at certain levels due to equipment limits or governor outputs with limited max power. Other examples are event protective devices, such as relays and breakers, where the range of allowed currents, voltages, or frequencies is limited to some domain and exceeding the range requires a change in the model being evaluated either by capping the output or otherwise altering the model equations in some fashion.

Unscheduled events refer to points in time when a component $x_i$ hits the boundary of the acceptable range, $x_i^{\text{min}}$, or $x_i^{\text{max}}$. To include these boundaries in the power system model, typical time-sequential simulations alter the right-hand side of the corresponding ODE model by requesting $\dot{x}_i = 0$ after an unscheduled event happened. The simulation continues with this zero right-hand-side leaving $x_i$ constant, until the remaining system components change in such a way that the original right-hand-side, $f_i$, pushes $x_i$ back to the interior of the acceptable range. Power system simulations handle these conditions either by evaluating the limits at each time point for fixed time step solvers or using root finding methods for adaptive time integration algorithms.

A. Adaptive time integration and unscheduled event handling

Unscheduled events can cause sudden changes and discontinuities in the transient power system simulation. In order to efficiently adapt to these changes, adaptive time-stepping methods solve the discretized system of DAES by marching forward in time in a step-by-step manner, while monitoring the local truncation error of the current step, in order to determine suitable step sizes that limit the local error below a certain threshold. For Runge-Kutta methods of order $q$ for example (as used in the numerical tests below), the next step size $h^{n+1} = t^{n+1} - t^n$ is determined to satisfy

$$\left(\frac{h^{n+1}}{h^n}\right)^{q+1} \|y^n - \tilde{y}^n\| \leq \epsilon$$

for a given tolerance $\epsilon > 0$, where $y^n$ and $\tilde{y}^n$ are two approximations from Runge-Kutta schemes of order $q$ and $q + 1$, respectively, and $y = (x, z)$.

Adapting the time-step size enables one to capture and march to the near exact time of an unscheduled event, and then restart the integration from there using the modified right-hand-side. In order to detect the event location, adaptive time-serial solvers define root finding functions that change sign if the solution results in a domain change. For example, for the equipment limit $x_i \in [x_i^{\text{max}}, x_i^{\text{min}}]$, the root-function component can be

$$r_i(x) = \min(x_i^{\text{max}} - x_i, x_i - x_i^{\text{min}}).$$

Each step forward in time is followed by a check for a sign change of each component of $r$. If any of the components indicates that a root has been crossed, the exact time of that sign change is determined together with the system state at that root point (e.g. through bisection or interpolation). This procedure enables highly accurate solutions even in the presence of discontinuities [10].

III. Parallel-in-Time Method

We apply the multigrid reduction in time (MGRIT, [6]) algorithm to parallelize across the time-domain of the simulation. MGRIT applies a nonlinear multigrid scheme (Full Approximation Storage, FAS, [11]) to the time domain of the discretized set of DAES.

Consider a discretization of the time domain $[0, T]$ with $0 = t^0 < t^1 < \cdots < t^N = T$, and let $y^n = y^n(t^n)$ denote the discrete approximation to the system state at time $t^n$, with $y = (x, z)$. Consider a time step function $\Phi$, that propagates a discrete state forward in time with

$$y^n = \Phi^n(y^{n-1}), \quad n = 1, \ldots, N$$

starting from an initial condition $y^0 = y(0)$, e.g. $\Phi^n$ can represent an (implicit or explicit) Runge-Kutta time-integration scheme that propagates the state $y^{n-1}$ to $y^n$, using a step size $h^n$.

While sequential time-marching is an optimal $O(N)$ method, it is inherently serial along the time domain such that finer time discretizations, or longer time horizons, result in equally larger time-to-solution. In contrast, MGRIT solves the above set of equations (6) simultaneously with an iterative multigrid scheme, aiming to achieve nearly constant run times when increasing the time domain length and compute resources.

To that end, a hierarchy of time grid levels is created by assigning every $m$-th point to a coarser time grid. This assignment results in a partitioning of the time points into so-called F- and C-points. Each multigrid iteration cycles through the different time-grid levels (with e.g. V-cycles, F-cycles, see e.g. [11] for various cycling strategies) while applying smoothing operations to update F- and C-points in parallel. The smoothing operation on a current time grid updates F- and C-points by propagating forward the corresponding left-neighbouring F- and C-point states, respectively. Typically FCF-relaxation is applied, which refers to a successive sweep over F-, then C-, then F-points again, using a coarse-grid approximation to the propagator $\Phi$ and a correction term involving the residual between successive time-grid levels. The coarse-grid approximation is a re-discretization of $\Phi$ to the current time-grid by choosing the correspondingly bigger time-step size associated with that time grid. Smoothing is a highly parallel operation as F- and C-relaxation sweeps are applied locally on each time interval and independently from each other. Serial forward propagation is limited to the size of the coarsest grid only. States are simply copied to transfer values between the time grid levels (no averaging is done).
At convergence, MGRIT reproduces the same trajectory as the time-sequential simulation (up to a tolerance). However, MGRIT enables greater concurrency along the time-domain. This concurrency creates a cross over point in terms of computational resources after which the parallel MGRIT approach provides speedup over the time-sequential simulation.

The open-source software library XBraid [12] implements the MGRIT iterations for existing time-integration schemes and provides the software infrastructure for passing data between grids and distributed parallelization with MPI. XBraid is agnostic to the underlying time-integrator \( \Phi \) and can provide an easy transition from existing time-sequential codes to a time-parallel MGRIT solver. XBraid mainly requires the user to wrap the action of the time-propagator into a separate \textit{Step} routine that can execute \( \Phi \) for various time-step sizes, \( h^n \), given from the current time grid.

In [9], the authors incorporate adaptive time steps into the MGRIT iterations through successive temporal refinement. Refinement occurs on the currently finest time grid, based on the local truncation error measure and resulting suggested step size reported from the time-stepping routine, \( \Phi \), as in (4). Utilizing the suggested new time-step size \( h^{n+1} \), the corresponding interval is uniformly refined into

\[
rfactor = \left\lfloor \frac{h^n}{h^{n+1}} \right\rfloor
\]

sub-intervals, if \( h^{n+1} < h^n \). After each MGRIT iteration, the reported refinement factors are used to create a new, finer grid for the next MGRIT iteration, see Figure 1.

![Temporal refinement with MGRIT](graphic.png)

**Fig. 1.** Temporal refinement with MGRIT. Graphic taken from [7].

### A. Unscheduled events

In order to handle unscheduled events in the time-parallel MGRIT solver, modifications to the existing approach that capture the exact time points of the events and thereby allow for accurate integration around unscheduled discontinuities are necessary. The goal is to let MGRIT create a time-grid such that (a) the local error measure from \( \Phi \) is smaller than the user defined tolerance \( \epsilon \), (b) no time-grid interval contains more than one event, and (c) the total number of time points is minimized to some extent.

To achieve this goal, our approach is twofold: we utilize the adaptive error measure from \( \Phi \) to report refinement factors to be used within MGRIT. Then, we utilize the root-finding technique in order to detect an unscheduled event within a current step, and if found, perform a second step from the root point to the end point of the interval. Due to the iterative nature of the MGRIT approach, it can not be guaranteed that every step forward in time starts with the state in the acceptable range. Thus, the root-finding technique as described for the time-serial case needs to be complemented with a limiting approach to enforce the equipment limits.

To be more precise, the \textit{Step} - routine of the MGRIT scheme on a current time-grid level \( l \) propagates states from \( t^n \) to \( t^{n+1} \) as follows:

1. Limit \( y^n_t \) back to the acceptable range for all component limits \( i \).
2. Integrate forward in time \( y^{n+1} = \Phi(y^n) \), utilizing the step size \( t^{n+1} - t^n \) on the current grid \( l \).
3. If grid \( l \) is currently the finest time grid:
   a) If a root has been crossed in \([t^n, t^{n+1}]\), find the exact time of the root point \( t^{\text{root}} \) with the root-finding technique and compute a consistent solution \( y^{\text{root}} \) with a nonlinear solve. Perform another step from \( t^{\text{root}} \) to \( t^{n+1} \) as \( y^{n+1} = \Phi(y^{\text{root}}) \) using the step size \( t^{n+1} - t^{\text{root}} \). If this step crosses a second root, refine by at least a factor of two based on the local error measure as in (7).
   b) Limit the point \( y^{n+1} \) back to the acceptable range.
   c) Refine the interval with the local error measure and refinement factor as in (7).

Limiting the components back to the limit boundaries requires solution of a system of nonlinear equations to ensure consistency of the remaining components at that time step.

In contrast to the time-serial simulation, the above approach never explicitly adds the time points of an unscheduled event to the current time grid but rather treats the event implicitly by performing a smaller step towards the root point and a second one starting from the root. Not adding the point explicitly to the current time grid prevents MGRIT from adding too many time points in the vicinity of an unscheduled event at the early stage of the execution, where the global error is still high and state information may still be far from convergence.

### IV. Results

In order to verify our approach, we set up a problem modelling a simple exciter component with equipment limits as in Figure 2. Nominally this functions has an amplifier with gain and output limits. For a given time-varying input voltage

\[
\begin{aligned}
V_{in}(t) \quad V_{\text{max}} \\
G_0 \quad V_{\text{min}} \\
V_{out}(t)
\end{aligned}
\]

**Fig. 2.** Exciter amplifier model problem with a given time-dependent input \( V_{in}(t) \), including equipment limits at \([V_{\text{min}}, V_{\text{max}}]\).

\[\dot{V}_{out}(t) = \frac{1}{\tau} \left( G_0 V_{in}(t) - V_{out}(t) \right), \]

(8)
if \( V_{out}(t) \) is inside the limits \( V_{min} \leq V_{out}(t) \leq V_{max} \), and \( V_{out}(t) = 0 \) otherwise. Here, \( G_0 > 0 \) is a given exciter gain factor, and \( \tau > 0 \) is a time constant. For the numerical results below, we choose \( G_0 = 2 \), and \( \tau = 0.1 \) seconds, and the input voltage being a superposition of two sinusoidal signals

\[
V_{in}(t) = c_1 \sin(2\pi f_1 t) + c_2 \sin(2\pi f_2 t)
\]  

(9)

with amplitudes \( c_1 = 2 \), \( c_2 = 1.5 \), and frequencies \( f_1 = 1 \), \( f_2 = 0.7 \). Limits are set to be \( V_{min} = -4 \) and \( V_{max} = 4 \). The test case is set up in such a way that roughly two unscheduled events occur per second. The time-dependent input and output voltages are shown in Figure 3.

![Exciter input and output signal over time](image)

Fig. 3. Exciter input and output signal over time, enforcing limits at \([-4, 4]\).

We discretize the ODE in time using a 4-th order implicit Runge-Kutta scheme [13]. We start the adaptive MGRIT scheme using a temporal grid with equally spaced time points with step size \( h^n = 0.2 \) and use the Runge-Kutta error measure to refine intervals as described above, with a tolerance of \( \epsilon = 10^{-6} \). For all test cases, we choose a coarsening factor of 5 to create a hierarchy of coarse time grid levels and limit the coarsest grid to contain no less than 20 points in time. We apply V-cycles in each multigrid iteration. For the parallel scaling tests, we utilize Quartz, an Intel Xeon-based cluster at Lawrence Livermore National Laboratory, with 2 sockets and 36 cores per node.

A. Adaptive MGRIT convergence

We first investigate MGRIT convergence for various time-domain lengths of up to \( T = 320 \) seconds, containing up to 640 unscheduled events. From Table I, we observe fast convergence of the time-adaptive MGRIT scheme that is independent of the time domain length and independent of the number of unscheduled events. The total number of time points on the finest time grid is at the same order for both the time-sequential scheme and the time-parallel MGRIT approach. MGRIT refines to a slightly finer grid, with an increase of fine-grid points of about 16% and a corresponding drop in the global error of one order of magnitude where error is measured against a finely resolved reference solution.

Figure 4 visualizes the benefit of not adding event points explicitly to the time grid during time-parallel MGRIT iterations, but instead treating unscheduled events implicitly by performing a second step in intervals that include an event. Adding event locations to the time-grid during MGRIT iterations results in very small step sizes around the time of the event because the current approximation to the event location can not be determined correctly until MGRIT has fully converged. In fact, this method results in 23% more points than a sequential run for the test problem above. For a problem with greater numbers of components with events, this increased number of points could be detrimental to efficiency. On the contrary, the implicit event treatment allows for varying event locations during the time-parallel solve, while maintaining accuracy.

![Step sizes of the adaptive time-serial simulation and the time-parallel MGRIT solver](image)

Fig. 4. Step sizes of the adaptive time-serial simulation and the time-parallel MGRIT solver, treating unscheduled events either implicitly by performing a double-step or explicitly by adding the event point to the current grid.

B. Parallel scaling of the time-parallel solve

We examine parallel scalability next by running strong and weak scaling studies. Figure 5 plots run times of the time-parallel MGRIT solver utilizing more and more resources for time-parallelization, compared to time-sequential simulations. Good strong scaling can be observed for MGRIT, where run times nearly reduced by half when doubling the number of cores used for time parallelization. The benefit of the time-parallel solver becomes more significant when longer time domains are considered, as time-sequential run times (horizontal lines) double when increasing the problem size. The MGRIT solver, however, produces nearly constant run times when increasing the problem size and the processor counts simultaneously proving good weak scaling properties. The corresponding run times and computed speedup of the time-parallel MGRIT solver over the time-sequential integration are

### Table I

<table>
<thead>
<tr>
<th>T (sec)</th>
<th>#iter</th>
<th>#refine</th>
<th>#time points on finest grid (global error)</th>
<th>sequential</th>
<th>time-parallel</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>2</td>
<td>3</td>
<td>5.116 (1.62e-5)</td>
<td>5.954 (1.84e-6)</td>
<td></td>
</tr>
<tr>
<td>80</td>
<td>2</td>
<td>3</td>
<td>10.230 (3.29e-5)</td>
<td>11.896 (3.59e-6)</td>
<td></td>
</tr>
<tr>
<td>160</td>
<td>2</td>
<td>3</td>
<td>20.457 (6.64e-5)</td>
<td>23.780 (7.10e-6)</td>
<td></td>
</tr>
<tr>
<td>320</td>
<td>2</td>
<td>3</td>
<td>40.910 (1.33e-4)</td>
<td>47.551 (1.40e-5)</td>
<td></td>
</tr>
</tbody>
</table>
reported in Table II, where a linear increase in speedup is observed for increasing time-domain length.

![Fig. 5. Run times of the MGRIT solver for various time domain length over compute resources. Horizontal lines indicate the corresponding sequential runtimes.](image)

This work was supported by the United States Department of Energy Office of Electrical Delivery and Energy Reliability Advanced Grid Modeling Program. This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344. LLNL-CONF-796422.

This document was prepared as an account of work sponsored by an agency of the United States government. Neither the United States government nor Lawrence Livermore National Security, LLC, nor any of their employees makes any warranty, expressed or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States government or Lawrence Livermore National Security, LLC. The views and opinions of authors expressed herein do not necessarily state or reflect those of the US government or Lawrence Livermore National Security, LLC, and shall not be used for advertising or product endorsement purposes.

### REFERENCES


