Received 29 April 2020; revised 30 June 2020; accepted 25 July 2020. Date of publication 29 July 2020;
date of current version 10 August 2020.

Digital Object Identifier 10.1109/OAJPE.2020.3012636

Parallel-in-Time Object-Oriented Electromagnetic Transient Simulation of Power Systems

**TIANSHI CHENG (Graduate Student Member, IEEE),
TONG DUAN (Graduate Student Member, IEEE),
AND VENKATA DINAVAHI (Fellow, IEEE)**

Department of Electrical and Computer Engineering, University of Alberta, Edmonton, AB T6G 2V4, Canada
CORRESPONDING AUTHOR: T. CHENG (tcheng1@ualberta.ca)

This work was supported by the Natural Science and Engineering Research Council of Canada (NSERC).

ABSTRACT Parallel-in-time methods are emerging to accelerate the solution of time-consuming problems in different research fields. However, the complexity of power system component models brings challenges to realize the parallel-in-time power system electromagnetic transient (EMT) simulation, including the traveling wave transmission lines. This paper proposes a system-level parallel-in-time EMT simulation method based on traditional nodal analysis and the *Parareal* algorithm. A new interpretation scheme is proposed to solve the transmission line convergence problem. To integrate different kinds of traditional EMT models, a component-based EMT system solver architecture is proposed to address the increasing model complexity. An object-oriented C++ implementation is proposed to realize the parallel-in-time Parareal algorithm based on the proposed architecture. The results on the IEEE-118 test system show 2.30x speed-up compared to the sequential algorithm under the same accuracy with 6 CPU threads, and a high parallel efficiency around 40%. The performance comparison of various IEEE test cases shows that the system's time-domain characteristics determine the speed-up of Parareal algorithm, and the delays in transmission lines significantly affect the performance of parallel-in-time power system EMT simulations.

INDEX TERMS Electromagnetic transient analysis, multi-core processors, object-oriented programming, parallel-in-time, parallel processing, power system simulation.

NOMENCLATURE

DAE	Differential-Algebraic Equation
DDE	Delay Differential Equation
EMT	Electromagnetic Transient
ODE	Ordinary Differential Equation
$\mathcal{F}$	Fine Solution Operator
$\mathcal{G}$	Coarse Solution Operator
$\boldsymbol{U}^k$	System State Vector of k-th Iteration
$\boldsymbol{W}$	System of Equations
Δt	Coarse-Grid Time-Step
δt	Fine-Grid Time-Step
n	n-th Discrete Simulation Time-Step
τ	Propagation Delay of Transmission Line

I. INTRODUCTION

THE electromagnetic transient (EMT) program, which simulates the temporary electromagnetic phenomena in the time domain such as voltage disturbances, surges, faults, and other transient behaviors in the power system, is essential for modern power system design and analysis [1]. The simulation often requires detailed models with high computation complexity to accommodate large-scale power systems. The algorithms of mainstream EMT tools are highly optimized using sparse matrix methods and fine-tuned power system models, but the performance is bound by the sequential programming based on the central processing unit (CPU). To get through the bottleneck, parallel computing became a popular option to speed up large-scale EMT simulation. By partitioning a large system into smaller parts, the rate of parallelism and the speed of convergence increased for nonlinear systems [2]. Both direct and iterative spatial domain decomposition were proposed to partition the system. Based on these domain decomposition methods, various parallel EMT off-line or real-time programs were implemented on different multi-core CPU, many-core graphic processing unit (GPU), field programmable gate array (FPGA), and multiprocessor systems-on-chip (MPSoCs) architectures [3]–[5]. However, few works were done on EMT simulation based on the parallel-in-time domain decomposition.

Parallel-in-time algorithms have a long history of 50 years [6], which are now widely used in many research fields to solve time-consuming simulation problems [7], [8]. In electrical engineering, parallel-in-time methods to solve power system dynamic problems were proposed in the 1990s [9]. After 1998, there had been few works on the parallel-in-time simulation until the recent 5 years. The Parareal algorithm, which solves the initial value problems iteratively using two ordinary differential equation (ODE) integration methods, has become one of the most widely studied parallel-in-time integration methods for its ability to solve both linear and nonlinear problems [6]. In 2016, [10] implemented a sequential program based on Parareal to solve the semi-explicit differential-algebraic equations (DAEs) system of power dynamic simulation problem and analyses the efficiency on theory. Reference [11] used the multi-grid reduction in time (MGRIT) method, which is a general form of Parareal, to solve a small 2-generator system in the full-implicit DAE form. Reference [12] utilized Parareal to solve partial differential equations of eddy current problems for finite element analysis (FEA). However, for EMT simulation, only a few research works about the parallel-in-time simulation of a single specific EMT model can be found. Reference [13] proposed a parallel-in-time algorithm that solves a simple average model of modular multilevel converter based on the Parareal iteration. The lack of research in EMT simulation is mainly due to: (1) Traditional time-domain simulation of electric circuits is based on nodal-analysis, which often yields a DAE with an index equal to or greater than one [14]. The implicit DAE system can only be solved by backward differentiation formulas (BDF), and many kinds of ODE-only parallel-in-time methods cannot be used. Although there are some works can convert DAE to state-space forms [15], it dramatically increases the matrix size compared to nodal analysis; (2) Although Parareal can solve nonlinear DAEs according to aforementioned research works, traditional EMT model implementations often have a fixed-step assumption, which needs adaptions to be used in Parareal; (3) In power systems, the most important components are transmission lines. In EMT simulation they are modeled with traveling wave models like the Bergeron line model, which brings delay differential equations (DDEs) to the system [16]. The dependency of past states creates additional convergence problems. No DDE was considered by previous parallel-in-time research works.

To address the aforementioned issues, this paper proposes a component-based system-level parallel-in-time power system EMT simulation algorithm based on the Parareal, which is implemented on the multi-core CPU with object-oriented C++ programming. Compared to other parallel-in-time research, the proposed simulation program has the following advantages and features:

1) Based on highly abstracted component class, the system architecture is flexible and extensible to integrate different kinds of traditional EMT models of power system equipment into the parallel-in-time algorithm and maintain all the advantages from nodal analysis;
2) Initial support for delay differential equations. With the modified interpolation strategy, the convergence speed increases so that transmission line models are able to work with the Parareal;
3) Reusing solver workers and workspace to reduce memory usage and decrease the overhead caused by object allocation in the Parareal iterations.

The paper is organized as follows: Section II describes the basics of the Parareal algorithm and implementations of EMT models, including the proposed modifications to transmission lines; Section III introduces the Parareal EMT simulation algorithm implemented upon the component-based circuit solver architecture; Section IV presents the case study and efficiency analysis; Section V is the conclusion.

II. PARALLEL-IN-TIME MODELING

There are different kinds of iterative and direct parallel-in-time methods, but only some of the iterative methods can solve nonlinear ODE problems. Parareal algorithm, which can be derived as a multi-grid method to solve ODE problems. However, it can be proved that the Parareal algorithm is not limited to the solution of ODE problems, and thus is suitable for parallel-in-time EMT simulation.

A. PARAREAL ALGORITHM

This algorithm decomposes the simulation time $[t_0, t_{end}]$ into N smaller sub intervals $I_k = [T_{j-1}, T_j]$, where start-time $T_0 = t_0$, and end-time $T_N = t_{end}$. At every time point T_j, the system has a unique solution for its state variables $\boldsymbol{U}_j$ produced by a fine solution operator $\mathcal{F}(T_j, T_{j-1}, \boldsymbol{U}_{j-1})$. So, for N time intervals, following nonlinear equations can be established:

$$\boldsymbol{W}(\boldsymbol{U}) := \begin{cases} \boldsymbol{U}_1 - \mathcal{F}(T_1, T_0, \boldsymbol{U}_0) = 0, \\ \boldsymbol{U}_2 - \mathcal{F}(T_2, T_1, \boldsymbol{U}_1) = 0, \\ \vdots \\ \boldsymbol{U}_{N-1} - \mathcal{F}(T_{N-1}, T_{N-2}, \boldsymbol{U}_{N-2}) = 0, \end{cases} \tag{1}$$

where $\boldsymbol{U}_0$ is determined as the known initial value.

The system (1) can be solved by Newton's method. For every $\boldsymbol{U}_j \in \boldsymbol{U} = \{\boldsymbol{U}_1, \boldsymbol{U}_2, \ldots, \boldsymbol{U}_{n-1}\}$, only two entries are nonzero. The individual update formula for each $\boldsymbol{U}_j$ is given by

$$\boldsymbol{U}_j^{(k)} = \mathcal{F}\big(T_j, T_{j-1}, \boldsymbol{U}_{j-1}^{(k-1)}\big) + \frac{\partial \mathcal{F}}{\partial \boldsymbol{U}}\big(T_j, T_{j-1}, \boldsymbol{U}_{j-1}^{(k-1)}\big)\left(\boldsymbol{U}_{j-1}^{(k)} - \boldsymbol{U}_{j-1}^{(k-1)}\right) \tag{2}$$

where k represents the iteration number.

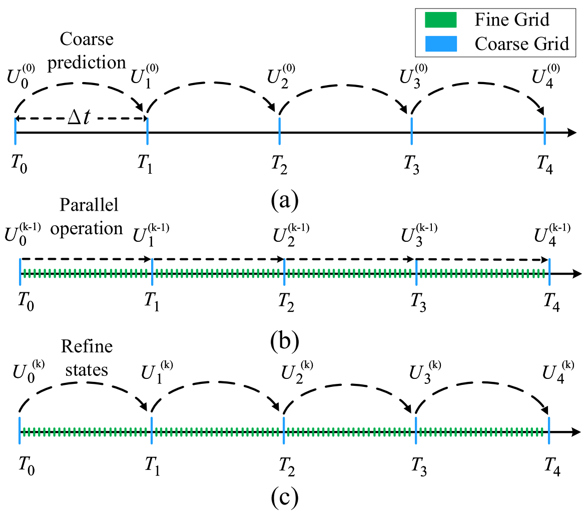


FIGURE 1. **Progression of steps in the Parareal algorithm: (a) Initialize $U_j^{(0)}$ which equals to $\mathcal{G}_j^{(0)}$; (b) Produce fine-grid solution $\mathcal{F}_j^k$; (c) Refine $U_j^{(k)}$ with $U_j^{(k)} = \mathcal{G}_j^{(k)} + \mathcal{F}_j^k - \mathcal{G}_j^{(k-1)}$.**

The $\frac{\partial \mathcal{F}}{\partial \boldsymbol{U}}$ item in (2) can be approximated by

$$\frac{\partial \mathcal{F}}{\partial \boldsymbol{U}}(T_j, T_{j-1}, \boldsymbol{U}_{j-1}^{(k-1)}) \approx \frac{\mathcal{F}(T_j, T_{j-1}, \boldsymbol{U}_{j-1}^{(k)}) - \mathcal{F}(T_j, T_{j-1}, \boldsymbol{U}_{j-1}^{(k-1)})}{\boldsymbol{U}_{j-1}^{(k)} - \boldsymbol{U}_{j-1}^{(k-1)}}. \quad (3)$$

Notice that (3) is differentiated by index k not j since it is the derivative of $\boldsymbol{U}$. To parallelize the computation, this item is approximated by a cheaper method $\mathcal{G}$ in sequential, which can produce a close approximation to $\mathcal{F}$, giving

$$\begin{aligned} \mathcal{F}(T_j, T_{j-1}, \boldsymbol{U}_{j-1}^{(k)}) &\approx \mathcal{G}(T_j, T_{j-1}, \boldsymbol{U}_{j-1}^{(k)}), \\ \mathcal{F}(T_j, T_{j-1}, \boldsymbol{U}_{j-1}^{(k-1)}) &\approx \mathcal{G}(T_j, T_{j-1}, \boldsymbol{U}_{j-1}^{(k-1)}). \end{aligned} \quad (4)$$

Substituting the $\mathcal{F}$ entries in (3) with the approximation in (4), following equation can be derived:

$$\begin{aligned} \boldsymbol{U}_j^{(k)} = \mathcal{F}\big(T_j, T_{j-1}, \boldsymbol{U}_{j-1}^{(k-1)}\big) \\ + \mathcal{G}(T_j, T_{j-1}, \boldsymbol{U}_{j-1}^{(k)}) - \mathcal{G}(T_j, T_{j-1}, \boldsymbol{U}_{j-1}^{(k-1)})\big), \end{aligned} \quad (5)$$

which becomes a Quasi-Newton method. This was first proposed in [17].

The Parareal algorithm can solve nonlinear full-implicit DAE problems in the condition of having a good approximation method $\mathcal{G}$ to predicted the states of $\mathcal{F}$ so that it can satisfy the assumptions in (4). The $\mathcal{G}$ and $\mathcal{F}$ method are called coarse and fine solution operator, and the time points they work at form up the coarse-grid and fine-grid, respectively. As shown in Fig. 1, the coarse operator makes initial predictions for the fine-grid; then the fine operator takes $\boldsymbol{U}$ as initial values and works in parallel to populate the fine-grid results; next, the coarse operator refines $\boldsymbol{U}$ states by making predictions based on new fine-grid solutions using (5). When the $\boldsymbol{U}$ stops changing, the fine-grid solutions are equal to the ones sequentially computed by the fine operator.

B. COMPONENTS MODELS

Normally, the nodal equations of a circuit have a DAE-index greater than zero [14], which means only implicit ODE methods can solve the equations. To solve the circuit with the nodal analysis method, the general form of linear circuit DAE discretized by Trapezoidal Rule can be reinterpreted as

$$\boldsymbol{Y}\boldsymbol{v}_{n+1} = \boldsymbol{s}_{n+1} - \boldsymbol{i}_{n+1}^{hist}, \quad (6)$$
$$\boldsymbol{i}_{n+1}^{hist} = g(\boldsymbol{x}_n, \boldsymbol{v}_n), \quad (7)$$
$$\boldsymbol{x}_{n+1} = h(\boldsymbol{i}_{n+1}^{hist}, \boldsymbol{v}_{n+1}), \quad (8)$$

where $\boldsymbol{Y}$ is the admittance matrix, $\boldsymbol{v}$ is node voltage vector; $\boldsymbol{s}_{n+1}$ is the source injection; $\boldsymbol{i}_{n+1}^{hist}$ is the equivalent current injections determined by (7); $\boldsymbol{x}$ is historical terms vector for components. The $\boldsymbol{x}$ and $\boldsymbol{v}$ form up the global system state vector $\boldsymbol{U} = \{\boldsymbol{v}, \boldsymbol{x}\}$. The nonlinear circuit has a varying $\boldsymbol{Y}$ matrix and additional current injections when the Newton-Raphson method is applied to linearize it at each time-step.

The equations are formed from different components in the power system. Most components in EMT simulation are treated as an equivalent conductance in parallel with a current source, which is convenient for nodal analysis. However, the behavior of these components differs very much from each other. It is more convenient to integrate their states on their local space, and the only common states they share are the node voltages.

1) INDUCTOR/CAPACITOR MODELS

These components can be discretized by Trapezoidal method in the general form [18]:

$$\begin{aligned} i_{n+1} &= G_{eq} v_{n+1} + i_{n+1}^{hist} \\ i_{n+1}^{hist} &= i_n + G_{eq} v_n. \end{aligned} \quad (9)$$

where

$$G_{eq}^L = \frac{2L}{\Delta t}, G_{eq}^C = \frac{\Delta t}{2C}. \quad (10)$$

G_{eq} is the discretized equivalent admittance for the inductor L or the capacitor C, i_{n+1} is the total current going through the inductor or capcitor component in the step $n+1$, and the i_{hist} is called *historical current source* which is injected to the right hand side of (7). The current i and v are the state variables of LC components.

In traditional fixed-step EMT simulation, the state variable i is substituted and omitted by current source i^{hist} because the G_{eq} remains unchanged between steps n and $n+1$, which gives:

$$i_{n+1}^{hist} = i_n^{hist} + 2G_{eq} v_n. \quad (11)$$

However, this only applies to fixed time-stepping. Since there are two different time-steps in Parareal algorithm, the form of (9) must be applied to all transient components. All fixed time-stepping assumptions must be avoided.

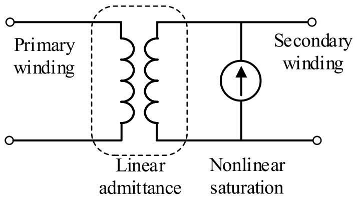


FIGURE 2. Admittance-based transformer model with saturation.

2) TRANSFORMER MODEL

The transformer for EMT simulation with winding resistance $\boldsymbol{R}$ and leakage inductance $\boldsymbol{L}$ is represented as

$$\boldsymbol{v} = \boldsymbol{R}\boldsymbol{i} + \boldsymbol{L}\frac{d\boldsymbol{i}}{dt}. \tag{12}$$

where $\boldsymbol{R}$ and $\boldsymbol{L}$ are $n \times n$ matrices; $\boldsymbol{v}$ and $\boldsymbol{i}$ are $n \times 1$ vectors of winding voltages and currents. Using Trapezoidal discretization, (12) can be written as

$$\begin{aligned} \boldsymbol{i}_{n+1} &= \boldsymbol{G}_{eq}\boldsymbol{v}_{n+1} + \boldsymbol{i}^{hist}_{n+1}, \\ \boldsymbol{i}^{hist}_{n+1} &= \boldsymbol{G}_{eq}\boldsymbol{v}_{n} - \boldsymbol{H}\boldsymbol{i}_{n}, \end{aligned} \tag{13}$$

where

$$\boldsymbol{G}_{eq} = (\boldsymbol{R} + \frac{2\boldsymbol{L}}{\Delta t})^{-1}, \boldsymbol{H} = \boldsymbol{G}_{eq}(\boldsymbol{R} - \frac{2\boldsymbol{L}}{\Delta t}).$$

The nonlinear saturation is modeled with a compensation current source i_s on the secondary winding. The nonlinear relationship between flux λ and saturation compensation current i_s is given by a nonlinear function $i_s = i_m(\lambda)$, details can be found in [18]. The λ is the integral of node voltage v over a time-step:

$$\lambda(t) = \lambda(t - \Delta t) + \int_{t-\Delta t}^{t} v(t)\, dt. \tag{14}$$

Transformer states $\{\boldsymbol{i}; \lambda\}$ are chosen to participate in the Parareal iteration.

3) GENERATOR MODEL

Indirect approaches are widely used in the EMT program to interface the synchronous machines. For example, the Norton current source representation of machines implemented in PSCAD/EMTDC®. However, this kind of model has weak numerical stability [19], and causes oscillations in Parareal iterations. Therefore, a machine model from [20], which can be used in variable time-stepping methods, is used in the proposed parallel-in-time simulation.

In this work, the synchronous generators are represented by the machine model with one *kd* winding and two *kq* windings.The relationship between voltages and currents can be expressed as:

$$\boldsymbol{v}_{um}(t) = \boldsymbol{R}_{um}\boldsymbol{i}_{um}(t) - \frac{d}{dt}\boldsymbol{\psi}_{um}(t) + \boldsymbol{u}(t) \tag{15}$$

$$\boldsymbol{\psi}_{um}(t) = \boldsymbol{L}_{um}\boldsymbol{i}_{um}(t), \tag{16}$$

where $\boldsymbol{v}_{um} = \left[v_d, v_q, v_0, v_f, 0, 0, 0\right]^{\mathrm{T}}$, $\boldsymbol{i}_{um} = \left[i_d, i_q, i_0, i_f, i_{kd}, i_{kq1}, i_{kq2}\right]^{\mathrm{T}}$, $\boldsymbol{\psi}_{um} = \left[\psi_d, \psi_q, \psi_0, \psi_f, \psi_{kd}, \psi_{kq1}, \psi_{kq2}\right]^{\mathrm{T}}$, $\boldsymbol{u} = \left[-\omega\psi_q, \omega\psi_d, 0, 0, 0, 0, 0\right]^{\mathrm{T}}$, $\mathbf{R}_{um} = \mathrm{diag}\left(R_d, R_q, R_0, R_f, R_{kd}, R_{kq1}, R_{kq2}\right)$ and $\mathbf{L}_{um}$ is the leakage inductance matrix.

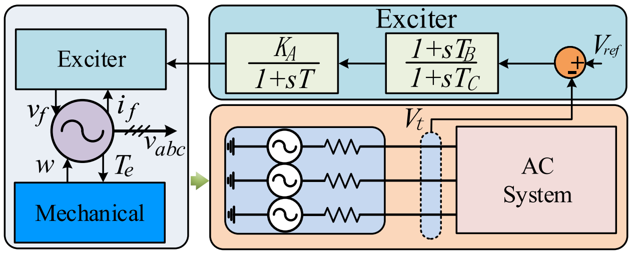


FIGURE 3. Synchronous generator representation using variable time-stepping machine model.

The machine is represented by a Thévenin voltage source and a resistance. Details can be found in [20]. Since the mechanical state ω changes slower than electrical ones, the numerical stability is better than the models that make relaxations or assumptions on electrical state variables. In the Parareal iteration, the state vector for this model is $\{\boldsymbol{v}_{um}, \boldsymbol{i}_{um}, \boldsymbol{\psi}_{um}, \omega\}$.

C. TRANSMISSION LINE MODEL

The transmission lines are the most common components in a power system. However, the models are based on the traveling wave theory which means that the solutions are dependent on a range of past states. This brings DDEs to the EMT power systems simulation [16].

Taking the lossless line as an example, the equations to update historical current source are given as:

$$\begin{aligned} i^{hist}_m(t) &= -2Gv_k(t-\tau) - i^{hist}_k(t-\tau) \\ i^{hist}_k(t) &= -2Gv_m(t-\tau) - i^{hist}_m(t-\tau). \end{aligned} \tag{17}$$

where τ is the transmission delay i^{hist}_m is the receiving-end current source, i^{hist}_k is the sending-end current source and G is the characteristic conductance of the transmission line. Details can be found in [18]. Since the equations are in continuous-time domain, to work with discrete-time integration, linear interpolation is used to get the approximation between two discrete time points near $t - \tau$. However, the traditional method cannot work under the parallel-in-time scenario for two reasons.

First, the DDE problems are not considered in previous parallel-in-time research. Although Parareal algorithm can solve nonlinear DAE problems by predicting states at certain time points in coarse-grid and refine them in fine-grid, it is difficult to do such thing for a transmission line because the historical states for the fine-grid transmission lines do not exist in coarse-grid. Using interpolation to predict the fine-grid history vectors cannot reflect the transient waveform of the discrete system with a smaller time-step. In this case, all transmission line history data should be prepared before Parareal iteration. Currently, limiting the time window of iteration is the only way to avoid this dependency issue.

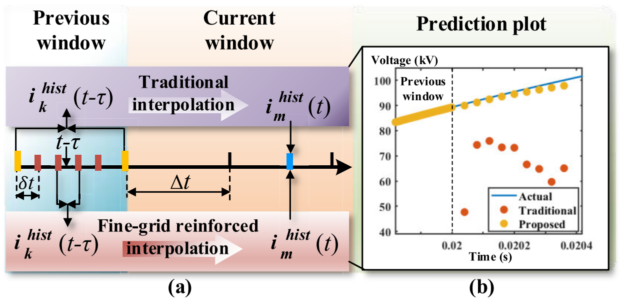


FIGURE 4. **(a) Proposed interpolation scheme; (b) Prediction results comparison.**

Second, limiting the time window is still not enough. The traditional transmission line uses linear interpolation to approximate the historical value at $t - \tau$. However, the approximated historical values are inconsistent with the fine-grid ones. As shown in Fig. 4 (b), the traditional interpolation's inconsistency causes a huge error between coarse and fine-grid so that the prediction in the next window fails to meet the assumption in (4) and causes deviations.

To solve these problems, a fine-grid reinforced transmission line model implementation is proposed, which is shown in Fig. 4 (a). Unlike conventional line models, where each line has its history vector to cover only one delay cycle, a transmission line in fine-grid and coarse-grid share the same memory for historical data. The computation is set to a time window which is smaller than the transmission delay τ, so that accurate historical data are prepared from the previously completed window, which avoids the data dependency issue. To improve the coarse-grid prediction, the coarse-grid transmission lines must read data from the data points in fine-grid history vectors, which requires a conversion between two time-steps to get the data index, given by

$$j_{fine} = floor(j_{coarse}\frac{\Delta t}{\delta t} + (\frac{\Delta t}{\delta t} - 1)\frac{\tau}{\Delta t}), \tag{18}$$

where j_{fine} is the converted index in fine-grid history vector for coarse-grid transmission line; j_{coarse} is the original index in Δt step simulation; δt is the fine-grid time-step, and the index is always truncated to an integer.

III. PARAREAL APPLICATION TO POWER SYSTEM EMT SIMULATION

Unlike other research works on parallel-in-time simulation, which focuses on specific differential equations, the power system EMT simulation needs to handle different kinds of equations and configurations. Also, Parareal algorithm requires the ability to restart the simulation at an arbitrary time to do iterations. Therefore, it is necessary to model the power system on a higher-level abstraction. A component-based system architecture is proposed to handle the parallel-in-time complexity flexibly and elegantly, and serves as the fundamental element to build the parallel-in-time simulation program.

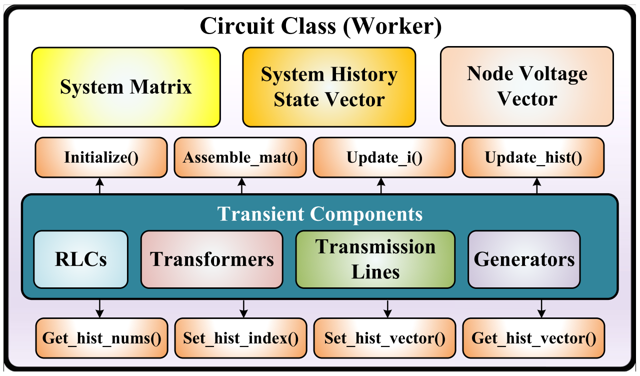


FIGURE 5. **Circuit class architecture for the proposed parallel-in-time EMT simulation program.**

A. COMPONENT-BASED SYSTEM ARCHITECTURE

A circuit is an undirected graph topological relationship between different components, where the components contain all the edge information of the graph. Therefore, the circuit can be represented with a vector of components. As shown in Fig. 5, the object-oriented concept is used to model the circuit system and components. Major properties of the *circuit* class contain the system matrix, state vectors, and a container for heterogeneous dynamic time-varying components.

All the components inherit from abstract base component class-*TransientComponent* so that the circuit object is able to call individual component functions with standard interface: *initalize(); assemble_mat(); update_i(); update_hist()*, with object polymorphism features. The *get_hist_nums(); set_hist_index(); set_hist_i(); get_hist_vector()* functions are interface to gather or scatter state variables between individual components to the system history state vector.

The algorithm flow chart is shown in Fig. 6. Before solving the circuit, each component initializes its variables and equivalent conductance according to the time-step. Then, they assemble the global system matrix according to their branch information in the graph. With the matrix formed, the circuit is ready to be solved. With this architecture, the complexity of power system EMT models is encapsulated into two functions of circuit class, the *init()* and *step()*. The component-based object-oriented architecture has the following advantages:

1) High flexibility: It separates the individual component model and system-level computation logic with the application of *TransientComponent*. That means users can focus on describing model behaviors without caring about the structure of the global system. It is easy to substitute models with different object-oriented interfaces. By correctly implementing parallel-in-time interfacing functions, the components automatically become available for the parallel-in-time solution.
2) High scalability: The *Circuit* class serves as a generic solver for different kinds of systems. The number of components and matrix sizes can be arbitrary.

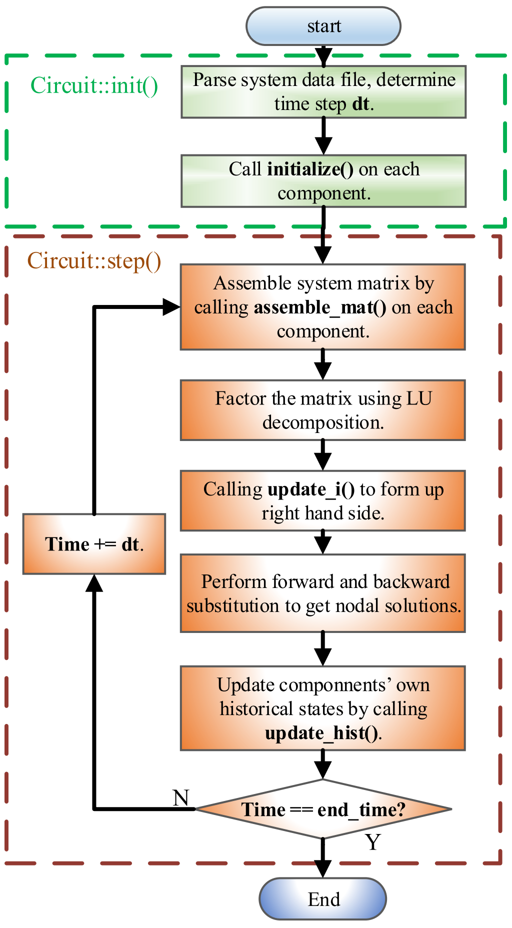

start
Circuit::init()
Parse system data file, determine time step **dt**.
Call **initialize()** on each component.
Circuit::step()
Assemble system matrix by calling **assemble_mat()** on each component.
Factor the matrix using LU decomposition.
Calling **update_i()** to form up right hand side.
Perform forward and backward substitution to get nodal solutions.
Update componnents' own historical states by calling **update_hist()**.
Time += dt.
Time == end_time?
N
Y
End

FIGURE 6. **Flowchart of proposed parallel-in-time EMT simulation program.**

For small systems, the implementation uses a single thread, which achieves the optimal performance, but for large-scale systems that have thousands of components, parallel class functions can be used to increase the performance and seamlessly integrated into the user's application code.

3) Modular design: The *Circuit* class is not only composed by decoupled modules but also designed as the basic building block of a parallel-in-time algorithm based on Parareal, which significantly reduces the implementation difficultly and human errors.

B. FIXED ALGORITHM IMPLEMENTATION

The basic version of Parareal algorithm is by using multiple circuit instances proposed in the previous section as workers. There is one coarse-grid worker, which is initialized with a larger time-step as a predictor, and many fine-grid workers with a smaller time-step; Their workspace is initialized before the beginning of computation; The coarse and fine-grid workers communicate with shared memory space. The algorithm working on a fixed full simulation time range is called the fixed Parareal algorithm in this paper. Suppose there are N coarse time intervals with initial time T_0 and state vector U_0, and k denotes the index number of Parareal iterations, then the algorithm is composed of the following 4 stages:

1) INITIALIZATION

The first coarse worker initializes with the system states $\boldsymbol{U}_0$ and $k = 0$ at $t = 0$; then the coarse worker generates initial guess with coarse time-step Δt and stores the N solutions into an array called $\boldsymbol{G}_k$ in the Fig. 7, where $k = 0$. This initial prediction process is only called once to start a full cycle of Parareal iterations in the designated simulation time range.

2) PARALLEL OPERATION

After $\boldsymbol{G}^{(k)}$ is prepared, the fine-grid workers load the initial solution according to their subdivision's position and initialize all variables in parallel. Notice that in each iteration, Only $N - k$ threads are launched to reduce overhead, since the first subdivision generates a solution which is guaranteed to converge.

After initialization, fine-grid workers work on their workspace to simulate m fine-grid steps, where $m \cdot \delta t = \Delta t$. The states we care about in Parareal iterations are located at the points on coarse-grid. The (m)th step states at each subdivision are extracted into the state vector $\boldsymbol{F}^{(k+1)}$ except the worker N because there is no (N)th state in coarse prediction. But this is not a problem because the (N)th state can converge once the $(N - 1)$th state is converged.

According to (5), it is possible to exploit the parallelism by merging the $\mathcal{F}^{(k+1)} - \mathcal{G}^{(k)}$ into the parallel operation stage. All workers execute $\boldsymbol{P} = \boldsymbol{F}^{(k+1)} - \boldsymbol{G}^{(k)}$, which is shown as red in the Fig. 7, in parallel after their solutions are done, which reduces the sequential overhead caused by the large-scale system. There is an exception that the $(k + 1)$th solution is already converged, so it is skipped and be used directly in this and next stage. After this step, all threads synchronize and wait for *SequentialUpdate* of coarse states.

3) SEQUENTIAL UPDATE

Now it is the time to perform $\boldsymbol{U}^{(k+1)} = \boldsymbol{G}^{(k+1)} - \boldsymbol{P}$ so that all states are corrected by new prediction. Obviously $\boldsymbol{G}_j^{(k+1)}$ in the Fig. 7, can only be processed in sequential depends on $\boldsymbol{U}_{j-1}^{(k+1)}$. The process starts from computing $\boldsymbol{G}_{k+2}^{(k+1)}$. There is no mistake to start with $(k+2)$th solution because the k state is already known and $(k+1)$th state is guaranteed to converge. So the square with text $(k + 1)$th is loaded into coarse worker first and produce $\boldsymbol{G}_{k+2}^{(k+1)}$. then $\boldsymbol{U}_{k+2}^{(k+1)}$ is computed so that it is ready to use $\boldsymbol{U}_{k+2}^{(k+1)}$ to compute $\boldsymbol{U}_{k+3}^{(k+1)}$, etc. Finally, all $\boldsymbol{U}_{k+1}^{(k+1)}$ to $\boldsymbol{U}_{N-1}^{(k+1)}$ are updated, and it is the time to check the convergence. This is done by

$$error = \sum_{j=k+1}^{N-1} \frac{\|\boldsymbol{U}_j^{(k+1)} - \boldsymbol{U}_j^{(k)}\|}{\|\boldsymbol{U}_j^{(k)}\|}. \tag{19}$$

Which computes the system states' relative Euclid distance between $(k + 1)$th and (k)th generation at each coarse time

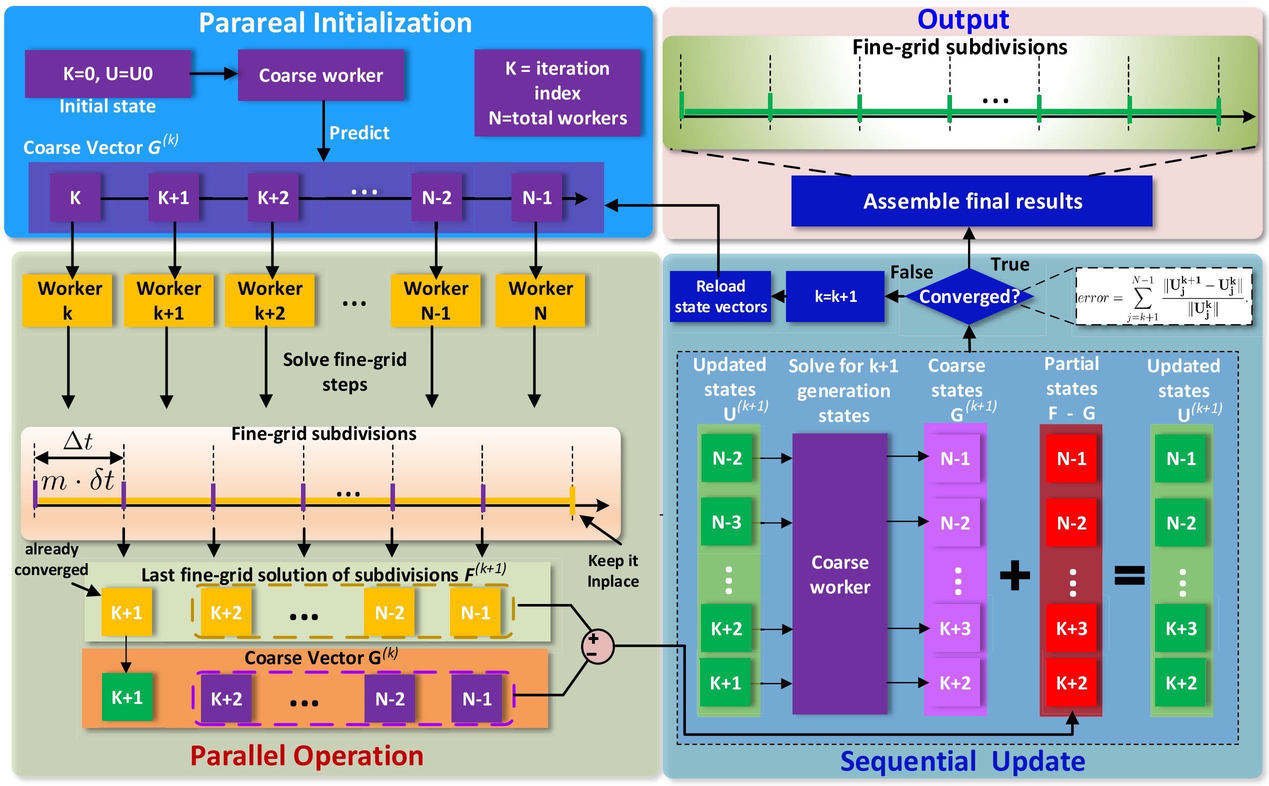


FIGURE 7. Detailed procedures in the proposed parallel-in-time EMT simulation.

point and sums them up. If the error is larger than the tolerance, the index (k)th increases by one and goes back to the *parallel operation* with newly updated states. If the computation converged, it is the time to generate final result.

4) OUTPUT

In this stage, all fine-grid workers use the converged coarse stated to fill all fine-grid steps in the output solution vector, which is the assembly procedure in the Fig. 7.

C. WINDOWED ALGORITHM IMPLEMENTATION

The fixed algorithm would waste much computation effort to work on a long time range for large systems, especially when there are many transmission lines, which require previous historical data to propagate traveling waves. Additionally, the memory consumption for a fixed algorithm is unacceptable since it needs the full-length memory allocation to work.

Therefore, a windowed version is adapted from the fixed algorithm. The windowed algorithm only launches several parallel workers and work on a small time window near the start point. To keep the implementation simple, there is no overlap between two consecutive windows. The workers' group first starts working on a known initial state and workspace. Once the solution for the current window is finished, the worker in charge of the last subdivision transfers the data to the first worker and all worker's simulation time and indices are reset for the next time window. Fig. 8 shows

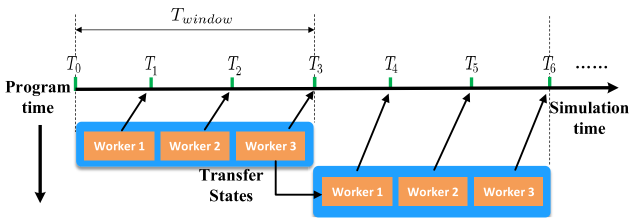


FIGURE 8. Example of windowed algorithm with 3 parallel workers.

an example of how the algorithm works, the *simulation time* is the time for system simulation, and the *program time* is the actual time consumed by executing. The convergence and parallel efficiency increase a lot in this way. Moreover, the memory allocation for the windowed algorithm can be limited to one time window.

IV. CASE STUDIES

A CPU-based parallel-in-time EMT program is implemented in C++ with Intel® Threading Building Block (Intel® TBB) and Intel® Math Kernel Library (Intel® MKL). Tests are performed on the IEEE-9, IEEE-39 and IEEE-118 test systems to verify the parallel-in-time results against the sequential ones. In addition, the parallel-in-time performance is compared to a traditional spatial parallel computing implementation which

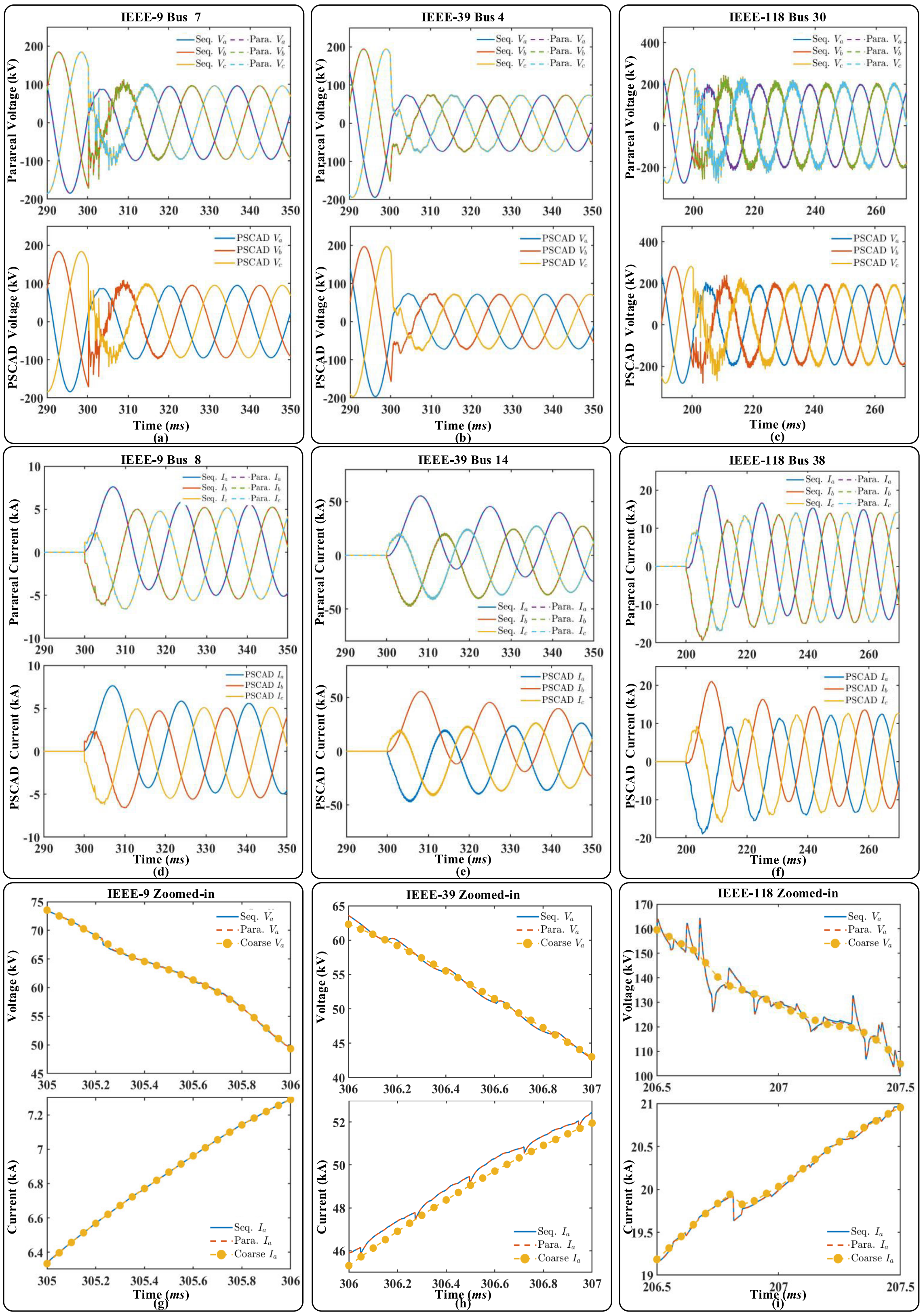


FIGURE 9. **Simulation results of three-phase voltages: (a) Bus 7 in IEEE-9; (b) Bus 4 in IEEE-39; (c) Bus 30 in IEEE-118. Simulation results of fault currents: (d) Bus 8 in IEEE-9; (e) Bus 14 in IEEE-39; (f) Bus 38 in IEEE-118. Zoomed-in comparison: (g) voltages and currents of IEEE-9; (h) voltages and currents of IEEE-39; (i) voltages and currents of IEEE-118.**

TABLE 1. Performance Comparison of Different Test Cases with Fixed $T_{window} = 200\mu s$, $\Delta t = 40\mu s$ and $\delta t = 1\mu s$ for A $500ms$ Duration Using 5 Threads.

	IEEE-9	IEEE-39	IEEE-118
Parareal (s)	4.019	25.233	283.393
Parallel LU (s)	4.756	46.641	385.024
Sequential (s)	4.507	44.579	591.015
Parareal Speed-up	1.12	1.77	2.09
Theortical Speed-up	1.88	2.18	2.24

utilized Intel® MKL's highly optimized parallel lower–upper (LU) decomposition algorithm.

The parameters of test cases are from [21]. The same thread number and algorithm configuration are used to compare the performance under the same condition. The time window and other parameters are shown in Table 1. For the IEEE-9 system, the minimal delay of transmission lines is $246\mu s$, which is enough for the $200\mu s$ time window. But for the other test systems, there are some short transmission lines. The transmission line length in IEEE-39 is scaled-up to 60-100km so that it can fit the time window and get reasonable speed-up. For the IEEE-118 case, the transmission lines below 60km are simplified to multiple-PI sections and the remaining 61 lines are modeled with the Bergeron model. Without the simplification, although good results can be obtained, the parallel-in-time algorithm falls back to the sequential program.

A three-phase-to-ground fault happens at 0.3s, Bus 8 in IEEE-9, and 0.3s, Bus 14 in IEEE-39 case. A fault happens at 0.2s in the IEEE-118 case at Bus 38. The fault resistor size is $R_{fault} = 0.01\Omega$, $R_{clear} = 1M\Omega$. To achieve stable and correct results, the error tolerance of a whole time window is set to 0.01, which means the relative error sum of coarse steps cannot exceed 1%. Parallel workers are fixed to five for parallel-in-time cases, and they use the sequential LU algorithm.

As shown in Fig. 9, the parallel-in-time simulation results coincide with those from the traditional method, and the zoomed-in views show the expected high accuracy. The results are verified with PSCAD/EMTDC®.

To evaluate the performance, the theoretical workload is defined by the simulation time consumption without any overhead from Parareal iterations or thread synchronizations. To achieve speed-up, the parallel-in-time workload must be smaller than the sequential one to get speed-up. The workload ratio can be obtained by T_{par}/T_{seq}, where T_{par} is the parallel workload and T_{seq} is the sequential workload, and the theoretical speed-up is the reciprocal of workload ratio. Also, the parallel efficiency is computed to evaluate the utilization of parallel processors. The efficiency is the ratio of the speed-up to the number of threads.

Normally a small scale system cannot benefit from parallel computing due to the thread launching and synchronization overhead. As shown in Table 1, the Parallel LU cannot achieve speed-up under the IEEE-9 (matrix size 27×27) and IEEE-39 (matrix size 117×117) case, while in IEEE-118 (matrix size 354×354) case the speed-up is obvious.

TABLE 2. Comparison of Sequential, Parallel LU, and parallel-in-time IEEE-118 Simulation with Various T_{window}, Δt and fixed $\delta t = 1\mu s$ for a $300ms$ Duration Using 5 Threads.

Configuration	Average iteration	Simulation time (s)	Speed-up	Theoretical speed-up	Eff.
$400\mu s, 80\mu s$	2.98	236.783	1.50	1.59	30%
$200\mu s, 40\mu s$	1.99	170.034	2.08	2.24	42%
$125\mu s, 25\mu s$	1.96	180.244	1.97	2.10	39%
$50\mu s, 10\mu s$	1.00	218.468	1.62	1.74	32%
Parallel LU	-	230.530	1.54	-	31%
Sequential	-	354.108	1.00	-	-

In contrast, the Parareal algorithm can get speed-up in all three cases because the speed-up is mainly determined by the temporal factors, which is more obvious in Table 3.

Table 2 shows the performance of the IEEE-118 simulation with the same fine-grid time-step $\delta t = 1\mu s$ and different coarse-grid time-steps. The parallel-in-time results are compared to sequential, parallel LU, and theoretical speed-up. The sequential and parallel LU simulation uses the same time-step $\delta t = 1\mu s$. All parallel-in-time simulation's error tolerances are set to 0.01 per time window so their results are on the same accuracy level.

From Table 2, the best performance case is when the time window is $200\mu s$. In this case, it is 2.08x faster than the sequential one while the parallel efficiency is 42%. The speed-ups are close to the theoretical ones and the overhead is around 10%, indicating that the implementation is highly efficient. The actual speed-ups and parallel efficiency are higher than the MKL parallel LU case except for the $400\mu s$ case.

When the $\Delta t = 80\mu s$, the theoretical speed decreases. This is understandable as the minimum transmission line delay is $200\mu s$. The $400\mu s$ time window exceeds the limit a lot so that the transmission lines cannot get historical data at the beginning, so it requires more iterations to converge. When $\Delta t = 10\mu s$, each fine-grid worker only takes 10 steps and the coarse worker takes 5 steps, which means the minimum workload ratio is $(5 + 10 + 4 + 10)/50 = 0.58$. In this way, although it can converge in one Parareal iteration, the efficiency is limited by the workload distribution between coarse and fine-grid. Therefore, appropriate time-steps and worker assignment are significant to achieve practical speed-up using the parallel-in-time method.

Table 3 shows the performance of all test cases with different thread numbers. The Δt, and δt remain constant for all thread numbers and test cases, so the time window size changes with the thread number. The different test cases show similar theoretical speed-ups. If the coarse prediction can match the fine-grid results on a wider range, the speed-up can be higher. However, in power system EMT simulations, the delay of transmission lines restricts the time window size so utilizing more threads may not get better performance. The exception is the IEEE-9 case, which gets the best theoretical speed-up with 16 threads. In general, the speed-up of the Parareal method is mainly affected by the system's

TABLE 3. Performance Comparison of Various Thread Number with Fixed $\Delta t = 40\mu s$ and $\delta t = 1\mu s$ for A $500ms$ Duration.

Thread	Theoretical Speed-up			Simulation Time (s)			Actual Speed-up		
Number	IEEE-9	IEEE-39	IEEE-118	IEEE-9	IEEE-39	IEEE-118	IEEE-9	IEEE-39	IEEE-118
16	2.27	1.43	1.93	8.63	52.79	480.76	0.52	0.84	1.23
12	1.87	1.74	1.91	7.52	39.26	408.86	0.60	1.13	1.45
8	2.18	1.74	1.83	4.40	34.50	340.75	1.03	1.29	1.73
6	2.14	2.21	2.55	3.77	25.16	256.80	1.20	1.76	2.30
5	2.14	2.18	2.24	3.63	25.23	287.95	1.25	1.76	2.05
4	1.79	1.79	1.83	4.08	29.52	334.33	1.11	1.50	1.77

time-domain characteristics such as the model's time constants and simulation time-steps rather than the system's scale.

Although the time-domain characteristics are dominant, the overheads of multi-thread synchronization and error evaluations are noticed in the smallest test case. The actual speed-up is much far away from the theoretical ones in the IEEE-9 case compared to larger cases, indicating much room for improvement in small scale systems. When the thread number equals 16, the CPU cannot finish all the jobs in parallel so the speed-up goes down significantly for all cases. The optimal thread number for these cases is 5 or 6, which gives a time window of 200-240μs. This is exactly the minimal transmission line delay set for all the test cases. This indicates that the transmission line delays are the main factors to affect the best speed-up we can get. exceeding this boundary causes inaccurate predictions so the speed-up drops down. Therefore, the treatment of the delays in transmission lines is significant for parallel-in-time power system EMT simulation. For IEEE-118 case, the best speed-up is 2.30x and the parallel efficiency is 38.3%, which is still higher than parallel LU decomposition.

V. CONCLUSION

A component-based simulation class architecture is proposed to handle different models in power systems, which delivers high flexibility and scalability to implement the system level parallel-in-time algorithm. Major challenges to handle the delay differential equations brought by transmission line models in the parallel-in-time algorithm are analyzed and a modified model implementation is proposed to solve the problem. Using the proposed circuit solver class as basic workers, the parallel-in-time algorithm based on the Parareal is implemented using object-oriented C++. The case study shows accurate results compared to the sequential program, while better parallel speed-up and efficiency than the MKL parallel LU implementation are obtained, showing the great potential of accelerating power system EMT simulation. The performance test also shows the system's time-domain characteristics determine the speed-up of Parareal algorithm, especially the transmission line delay in power system EMT simulation.

REFERENCES

[1] M. D. O. Faruque *et al.*, "Real-time simulation technologies for power systems design, testing, and analysis," *IEEE Power Energy Technol. Syst. J.*, vol. 2, no. 2, pp. 63–73, Jun. 2015.

[2] N. Frohlich, B. M. Riess, U. A. Wever, and Q. Zheng, "A new approach for parallel simulation of VLSI circuits on a transistor level," *IEEE Trans. Circuits Syst. I, Fundam. Theory Appl.*, vol. 45, no. 6, pp. 601–613, Jun. 1998.

[3] Z. Zhou and V. Dinavahi, "Fine-grained network decomposition for massively parallel electromagnetic transient simulation of large power systems," *IEEE Power Energy Technol. Syst. J.*, vol. 4, no. 3, pp. 51–64, Sep. 2017.

[4] Z. Shen, T. Duan, and V. Dinavahi, "Design and implementation of real-time Mpsoc-FPGA-based electromagnetic transient emulator of CIGRÉ DC grid for HIL application," *IEEE Power Energy Technol. Syst. J.*, vol. 5, no. 3, pp. 104–116, Sep. 2018.

[5] N. Lin and V. Dinavahi, "Parallel high-fidelity electromagnetic transient simulation of large-scale multi-terminal DC grids," *IEEE Power Energy Technol. Syst. J.*, vol. 6, no. 1, pp. 59–70, Mar. 2019.

[6] T. Carraro, M. Geiger, S. Rörkel, and R. Rannacher, *Multiple Shooting and Time Domain Decomposition Methods*. Heidelberg, Germany: Springer, 2015.

[7] L. Baffico, S. Bernard, Y. Maday, G. Turinici, and G. Zérah, "Parallel-in-time molecular-dynamics simulations," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 66, no. 5, 2002, Art. no. 057701.

[8] A. Blouza, L. Boudin, and S. M. Kaber, "Parallel in time algorithms with reduction methods for solving chemical kinetics," *Commun. Appl. Math. Comput. Sci.*, vol. 5, no. 2, pp. 241–263, Dec. 2010.

[9] M. La Scala, G. Sblendorio, and R. Sbrizzai, "Parallel-in-time implementation of transient stability simulations on a transputer network," *IEEE Trans. Power Syst.*, vol. 9, no. 2, pp. 1117–1125, May 1994.

[10] G. Gurrala, A. Dimitrovski, S. Pannala, S. Simunovic, and M. Starke, "Parareal in time for fast power system dynamic simulations," *IEEE Trans. Power Syst.*, vol. 31, no. 3, pp. 1820–1830, May 2016.

[11] M. Lecouvez, R. D. Falgout, C. S. Woodward, and P. Top, "A parallel multigrid reduction in time method for power systems," in *Proc. IEEE Power Energy Soc. Gen. Meeting (PESGM)*, Jul. 2016, pp. 1–5.

[12] S. Schops, I. Niyonzima, and M. Clemens, "Parallel-in-time simulation of eddy current problems using parareal," *IEEE Trans. Magn.*, vol. 54, no. 3, pp. 1–4, Mar. 2018.

[13] S. Debnath, "Parallel-in-time simulation algorithm for power electronics: MMC-HVdc system," *IEEE Trans. Emerg. Sel. Topics Power Electron.*, early access, Oct. 22, 2019, doi: 10.1109/JESTPE.2019.2947411.

[14] M. Günther and U. Feldmann, "The DAE-index in electric circuit simulation," *Math. Comput. Simul.*, vol. 39, nos. 5–6, pp. 573–582, 1995.

[15] T. Martinez-Marin, "State-space formulation for circuit analysis," *IEEE Trans. Educ.*, vol. 53, no. 3, pp. 497–503, Aug. 2010.

[16] Y. Kuang, *Delay Differential Equations: With Applications in Population Dynamics*. New York, NY, USA: Academic, 1993.

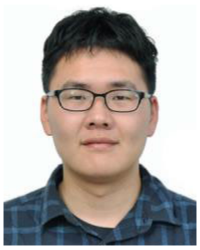
[17] M. J. Gander and E. Hairer, "Nonlinear convergence analysis for the parareal algorithm," in *Domain Decomposition Methods in Science and Engineering XVII*. Berlin, Germany: Springer, 2008, pp. 45–56.

[18] H. W. Dommel, *EMTP Theory Book*. Vancouver, BC, Canada: Microtran Power System Analysis Corporation, 1996.

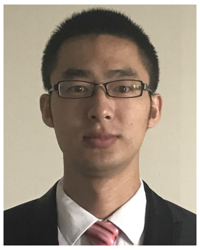
[19] L. Wang *et al.*, "Methods of interfacing rotating machine models in transient simulation programs," *IEEE Trans. Power Del.*, vol. 25, no. 2, pp. 891–903, Apr. 2010.

[20] T. Duan and V. Dinavahi, "Adaptive time-stepping universal line and machine models for real time and faster-than-real-time hardware emulation," *IEEE Trans. Ind. Electron.*, vol. 67, no. 8, pp. 6173–6182, Aug. 2019.

[21] S. Peyghami, P. Davari, M. Fotuhi-Firuzabad, and F. Blaabjerg, "Standard test systems for modern power system analysis: An overview," *IEEE Ind. Electron. Mag.*, vol. 13, no. 4, pp. 86–105, Dec. 2019.



TIANSHI CHENG (Graduate Student Member, IEEE) received the B.Eng. degree in electrical engineering and its automation from Southeast University, China, in 2017. He is currently pursuing the Ph.D. degree in electrical and computer engineering with the University of Alberta, Canada. His research interests include power systems analysis and simulation, power electronics, and high-performance computing.



TONG DUAN (Graduate Student Member, IEEE) received the B.Eng. degree in electronic engineering from Tsinghua University, Beijing, China, in 2013. He is currently pursuing the Ph.D. degree in electrical and computer engineering with the University of Alberta, Edmonton, AB, Canada. His research interests include real-time simulation of power systems, communication networking, parallel computing, and field-programmable gate arrays.

VENKATA DINAVAHI (Fellow, IEEE) received the B.Eng. degree in electrical engineering from the Visvesvaraya National Institute of Technology (VNIT), Nagpur, India, in 1993, the M.Tech. degree in electrical engineering from IIT Kanpur, Kanpur, India, in 1996, and the Ph.D. degree in electrical and computer engineering from the University of Toronto, Toronto, ON, Canada, in 2000. He is currently a Professor with the Department of Electrical and Computer Engineering, University of Alberta, Edmonton, AB, Canada. His research interests include real-time simulation of power systems and power electronic systems, electromagnetic transients, device-level modeling, large-scale systems, and parallel and distributed computing.