Received 21 September 2020; revised 1 January 2021 and 29 April 2021; accepted 8 July 2021. Date of publication 13 July 2021; date of current version 28 July 2021. The review of this article was arranged by Editor Dr. Luigi Martirano.

Digital Object Identifier 10.1109/OJIA.2021.3096518

Hierarchical Linking-Domain Extraction Decomposition Method for Fast and Parallel Power System Electromagnetic Transient Simulation

TONG DUAN (Student Member, IEEE) AND VENKATA DINAVAHI (Fellow, IEEE)

Department of Electrical and Computer Engineering, University of Alberta, Edmonton, AB T6G 2V4, Canada

CORRESPONDING AUTHOR: TONG DUAN (e-mail: tduan@ualberta.ca)

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

ABSTRACT The linking-domain extraction (LDE) decomposition method is a new non-overlapping domain decomposition method for parallel circuit simulation. However, the original LDE method is inefficient in both the computational procedure and storage cost. In this work, a novel hierarchical LDE (H-LDE) method is proposed to further improve the LDE method, which leverages all the hidden features of LDE that are not exploited in the original work to perform a multi-level decomposition of power systems. The LDE-based matrix equation solution computation procedure is first proposed to eliminate the necessity of computing the entire matrix inversion, and then the multi-level computation structure is proposed for fast matrix inversion of the decomposed sub-matrices. The mathematical complexity of the H-LDE method is analyzed, which is used to derive the two principles for decomposing a power system. These principles can be applied on both parallel and sequential compute architecture. The 4-level LDE decomposition is applied on the IEEE 118-bus test power system and implemented in both sequential and parallel, which is used to verify the validity and efficiency of the proposed H-LDE decomposition method. The simulation results of various benchmark test power systems show that the proposed H-LDE method can achieve better performance than the classical LU factorization and sparse KLU method within a certain system scale.

INDEX TERMS Circuit simulation, domain decomposition, graphics processing unit, inverse matrix calculation, linear equation solver, parallel processing, power system simulation.

I. INTRODUCTION

Electromagnetic transient (EMT) circuit simulation is an important tool to study a power system's behaviour under transient-level faults. However, the simulation process slows down significantly when the circuit scale expands [1]. Therefore, parallel computing techniques are increasingly needed and involved in circuit simulation area, which always utilizes the domain decomposition to first decompose the large system into small subsystems and then handle the decomposed subsystems in parallel [2], [3]. In non-overlapping domain decomposition, each interface node belongs to an unique subsystem and thus the iteration computation can be avoided. For example, the Schur complement method is a matrix-based non-overlapping decomposition method used in EMT simulation [4]–[7], which is different from the latency-based non-overlapping decomposition method such as the transmission line modeling (TLM) method [8], [9] and latency insertion method (LIM) [10], [11].

The new linking-domain extraction (LDE) non-overlapping decomposition method [12] is similar to the Schur complement method, but it can find the general matrix inversion formulation of the circuit conductance matrix that can be expressed as the sum of a linking-domain matrix (LDM) and a diagonal block matrix (DBM). The efficiency of the LDE method is achieved by computing the inversion of the small decomposed block matrices (in serial or parallel, which occupies much smaller computational latencies compared to computing the whole matrix equation) and then assembling the inverted block matrices via a correction matrix to obtain the inversion of the entire conductance matrix. However, this process is not scalable due to several reasons: 1) For large-scale power systems, the conductance matrix is large and sparse, but the inverted matrix is usually dense, which costs much more storage than simply solving the matrix equations and makes the LDE method inapplicable. Therefore, the LU factorization [4] is used in EMTDC/PSCAD^{TM} [13] and fast sparse solutions such as the KLU [14] and NICSLU [15] methods are also widely applied in power system simulators such as SPICE [16]. 2) The original LDE method decomposes the conductance matrix once, which may also result in a large size of the decomposed block matrices and computing the inversion of the block matrices is also costly. Therefore, although the LDE method has a strong mathematical base, it seems that it can only achieve better performance than than classical solvers in simulating a very small-scale power system.

To improve the original LDE method, in this work, the hierarchical LDE (H-LDE) decomposition method is proposed, which utilizes all the hidden features of the LDE method to achieve an all-around improvement. First, the matrix equation solver computation procedure based on LDE is proposed, which avoids computing the entire matrix inversion so that the storage cost is reduced significantly; second, a multi-level decomposition structure is proposed to reduce the computational cost of inverting the decomposed block matrices. The approximate complexity of H-LDE decomposition is analyzed, based on which the two decomposition principles are presented to instruct the detailed decomposition configuration for a specific number of decomposition levels: before the last level, the decomposition does not need to find a balance between the sizes of the DBMs and LDM; and in the last level, the decomposition should take the balance between DBMs and LDM into consideration. The detailed decomposition logic depends on the power system topology at hand, the parallelism capabilities of the parallel platform used and the number of decomposition levels.

The IEEE 118-bus test system is used to verify the validity and efficiency of the proposed H-LDE decomposition performance, both the sequential and GPU-based parallel implementation are discussed. The performance of H-LDE is also compared with the LU factorization and KLU method in several standard benchmarks, which shows the H-LDE method is much more scalable than the original LDE method and could achieve better performance than the pure LU factorization. The computational time also shows that the H-LDE cost lower time than the sparse KLU solver within a certain system scale. The paper is organized as follows: Section II introduces the LDE method and improved LDE calculation procedure. Section III presents the proposed hierarchical LDE method. In Section IV, the GPU-based implementation for the test system is described, and the simulation results are presented in Section V. Finally in Section VI conclusions are drawn.

II. IMPROVED LINKING-DOMAIN EXTRACTION BASED DECOMPOSITION METHOD

LDE is a matrix-based decomposition method, which is able to obtain the general formulation of the matrix inversion and compute the inverse in parallel [12]. However, computing the entire matrix inversion may be costly and meaningless. In this section, the mathematical formulation of the LDE method is introduced, and the improved LDE calculation procedure is proposed to optimize the matrix equation solution.

A. LDE MATRIX DECOMPOSITION

Given a power system containing N nodes, a $N \times N$ conductance matrix $\mathbf{G}$ will be generated. Then $\mathbf{G}$ could be decomposed into two separate matrices: the diagonal block matrix $\mathbf{G}_d$ and the linking-domain matrix $\mathbf{L}$:

$$\mathbf{G} = \mathbf{G}_d + \mathbf{L} \tag{1}$$

The number of block matrices in $\mathbf{G}_d$ depends on the number of subsystems decomposed. Figure 1 illustrates the case of two decomposed subsystems: there are n_1 nodes in subsystem S1 (matrix $\mathbf{G}_1$) connecting with n_2 nodes in subsystem S2 (matrix $\mathbf{G}_2$) via a conductance or voltage source (note that if two interface nodes are connected via a current source then it will not be revealed in the conductance matrix). The linking-domain matrix $\mathbf{L}$ ($N \times N$) is composed of several matrices with all-zero elements and a small linking-domain $\mathbf{L}_s$ with size of $(n_1 + n_2) \times (n_1 + n_2)$. If the number of decomposed subsystems is larger than two, the linking-domain matrix will contain several small linking-domains ($\mathbf{L}_s^1, \mathbf{L}_s^2, \ldots, \mathbf{L}_s^m$), and these small linking-domains may not have a integral matrix format if the interface nodes are not continuous in node indexes.

For a common network, the linking-domain matrix $\mathbf{L}$ can be expressed as a transformation from a diagonal matrix Λ ($k \times k$), and the transformation matrix $\mathbf{C}$ is a rectangular matrix ($N \times k$) of which the element values are only equal to 1, -1, or 0.

$$\mathbf{L} = \mathbf{C}\Lambda\mathbf{C}^T \tag{2}$$

Note that k is the number of links connecting the decomposed matrices. Take the case of two decomposed subsystems as an example, as shown in Fig. 1(b)(c), Λ has k negative elements in diagonal, and $\mathbf{C}$ is composed of all-zero matrices and a small transformation matrix $\mathbf{C}_s$ with size of $(n_1 + n_2) \times k$. $\mathbf{C}_s$ can be regarded as two parts: the upper n_1 rows $\mathbf{C}_s^{(1)}$ and the lower n_2 rows $\mathbf{C}_s^{(2)}$; every column of $\mathbf{C}_s^{(1)}$ only has one element equal to -1, and the other elements are equal to 0; while every column of $\mathbf{C}_s^{(2)}$ only has one element equal to 1, and the other elements are equal to 0.

Then the general formulation of the inverse matrix of $\mathbf{G} = \mathbf{G}_d + \mathbf{L}$ could be found based on the Woodbury matrix identity [17]:

$$\mathbf{G}^{-1} = (\mathbf{G}_d + \mathbf{L})^{-1} = \mathbf{G}_d^{-1} - \mathbf{G}_d^{-1}\mathbf{C}\mathbf{Q}\mathbf{C}^T\mathbf{G}_d^{-1} \tag{3}$$

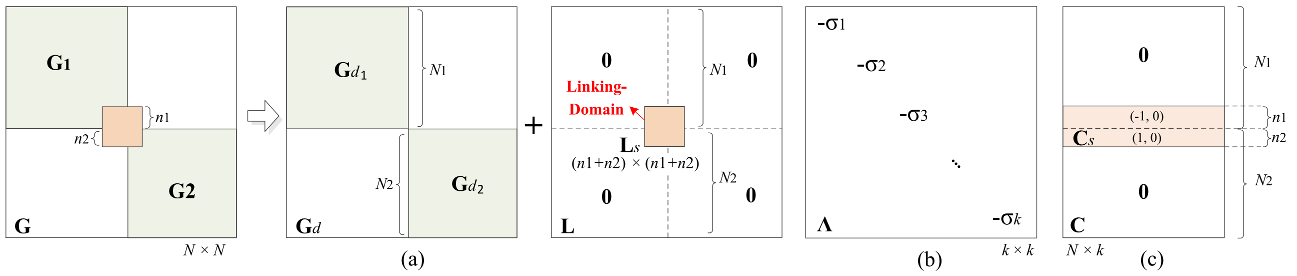


FIGURE 1. Example of LDE decomposition of two subsystems: (a) decomposition of G, (b) Λ matrix, and (c) transformation matrix C.

where:

$$\mathbf{Q} = (\Lambda^{-1} + \mathbf{C}^T \mathbf{G}_d^{-1} \mathbf{C})^{-1} \tag{4}$$

B. IMPROVED LDE COMPUTATION PROCEDURE

Although computing the matrix inversion $\mathbf{G}^{-1}$ directly could accelerate the simulation process significantly when the the conductance matrix $\mathbf{G}$ does not change over the simulation duration, the storage and I/O cost increase dramatically when the power system scale expands. Therefore, in this paper, the LDE computation procedure is improved accordingly.

The improved computational procedure is used to solve the matrix equations without storing the inverted conductance matrix. The goal is to solve for the node voltages v:

$$\mathbf{G}v = i_{eq} \tag{5}$$

Applying the LDE matrix inversion:

$$\begin{aligned} v &= \mathbf{G}^{-1} i_{eq} = [\mathbf{G}_d^{-1} - \mathbf{G}_d^{-1}\mathbf{C}\mathbf{Q}\mathbf{C}^T\mathbf{G}_d^{-1}] i_{eq} \\ &= \mathbf{G}_d^{-1} i_{eq} - \mathbf{G}_d^{-1}\mathbf{C}\mathbf{Q}\mathbf{C}^T\mathbf{G}_d^{-1} i_{eq} \\ &= v_{DBM} - \mathbf{G}_d^{-1}\mathbf{C}(\Lambda^{-1} + \mathbf{C}^T\mathbf{G}_d^{-1}\mathbf{C})^{-1}\mathbf{C}^T v_{DBM} \end{aligned} \tag{6}$$

where $v_{DBM} = \mathbf{G}_d^{-1} i_{eq}$ is the solution of each decomposed subsystems. The matrix inversion process of $(\Lambda^{-1} + \mathbf{C}^T\mathbf{G}_d^{-1}\mathbf{C})$ can also be avoided to reduce the computational and storage cost:

$$\begin{aligned} v &= v_{DBM} - \mathbf{G}_d^{-1}\mathbf{C}(\Lambda^{-1} + \mathbf{C}^T\mathbf{G}_d^{-1}\mathbf{C})^{-1}\mathbf{C}^T v_{DBM} \\ &= v_{DBM} - \mathbf{G}_d^{-1}\mathbf{C} v_{LDM} \end{aligned} \tag{7}$$

where v_{LDM} is the solution of the matrix equation below:

$$(\Lambda^{-1} + \mathbf{C}^T\mathbf{G}_d^{-1}\mathbf{C}) v_{LDM} = \mathbf{C}^T v_{DBM} \tag{8}$$

As stated in the matrix inversion procedure, $\mathbf{C}$ and $\mathbf{C}^T$ matrix have very special features, which enables the $\mathbf{G}_d^{-1}\mathbf{C}$ and $\mathbf{C}^T v_{DBM}$ to be obtained extremely simple without multiplication operations. Therefore, the improved LDE solver computation procedure can be executed as follows:

1) compute $\mathbf{G}_d^{-1}$ and $v_{DBM} = \mathbf{G}_d^{-1} i_{eq}$;
2) compute $\mathbf{T} = \mathbf{G}_d^{-1}\mathbf{C}$;
3) compute $\Lambda^{-1} + \mathbf{C}^T\mathbf{T}$ and solve v_{LDM};
4) compute the final solution $v = v_{DBM} - \mathbf{T}v_{LDM}$;

Using the above improved LDE computation procedure, the storage cost can be dramatically reduced. For a $N \times N$ conductance matrix that is decomposed into m sub-matrices with equal sizes, the storage cost of the improved LDE method is $O[m(N/m)^2 + k^2] = O(N^2/m + k^2)$, which is reduced nearly m times over the original LDE method. Although the storage is still large compared to the sparse LU factorization based solvers, the benefits of the fast solution process can be reflected within a certain power system scale.

III. HIERARCHICAL LDE METHOD

From the above procedure it can be observed that the computation of $\mathbf{G}_d^{-1}$ cannot be avoided although the computation of $\mathbf{G}^{-1}$ is eliminated. Since only the diagonal block matrices in $\mathbf{G}_d^{-1}$ are non-zeros, the storage cost can be reduced significantly. However, for large-scale power systems, the conductance matrix could not be decomposed in a fine-grained fashion because such decomposition will result in a large Λ matrix that is not beneficial to the overall performance. Therefore, the LDE decomposition will generate large block matrices in $\mathbf{G}_d$ even after decomposition, which makes the computation of $\mathbf{G}_d^{-1}$ also costly.

Fortunately, the LDE method is essentially a matrix inversion method, which is able to accelerate the matrix inversion process of the block matrices in $\mathbf{G}_d$. This means, although the LDE matrix inversion is not used in the improved computation procedure, it can also be used to compute $\mathbf{G}_d^{-1}$, and this application of LDE method is just to decompose the subsystems into further sub-subsystems. Based on this, the multi-level LDE decomposition is proposed, which is called "hierarchical LDE (H-LDE) method".

A. MULTI-LEVEL LDE DECOMPOSITION

When the decomposed subsystems also have relatively large scales, computing the inversion of block matrices in $\mathbf{G}_d$ still requires a lot of compute effort. In the H-LDE decomposition, the computation of $\mathbf{G}_d^{-1}$ could be executed based on the second or even higher level LDE decomposition to reduce the computational time. Thus the application of LDE method could be extended to simulate power systems in a hierarchical manner: the improved LDE computation procedure is applied for the first level LDE decomposition to reduce the storage and computation cost, because there is no need to compute the

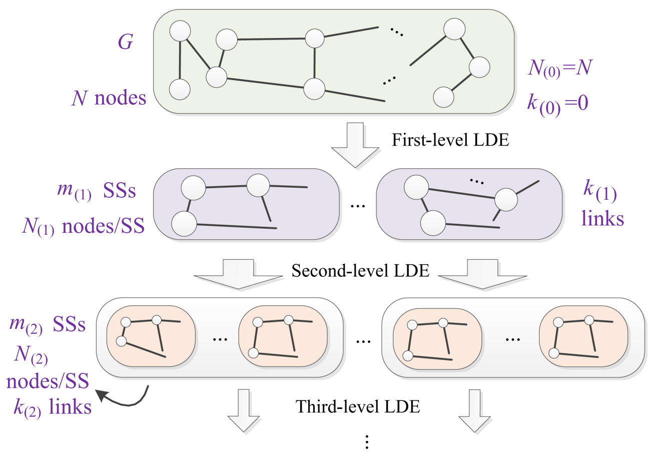


FIGURE 2. **Demonstration of hierarchical LDE decomposition.**

inverse of the whole matrix; then, the second or higher level LDE decomposition could be computed using the original LDE matrix inversion procedure, because the goal of multi-level LDE decomposition is to obtain $\mathbf{G}_d^{-1}$ quickly.

B. COMPUTATIONAL COMPLEXITY ANALYSIS OF HIERARCHICAL LDE

Assume the complexity of inverting a $N \times N$ matrix is $O(N^3)$. Then the original LDE method has a complexity of $O[N_j^3 + k^3]$, where N_j is the maximum size of the decomposed block matrices, and k is the number of links connecting these decomposed subsystems (SSs). However, this is not applicable for the H-LDE method, because complexity of computing $\mathbf{G}_d^{-1}$ should be re-evaluated. Assume that there are r levels of LDE decomposition in total, and the decomposed subsystems nearly have the same size while the number of links between the decomposed sub-subsystems are also the same for the decomposition of different subsystems. This assumption may be not rigorous, but considering that the power system topology is not dense and the connections are relatively distributed on average, this assumption is just an approximation and makes sense in the complexity analysis. As shown in Fig. 2, after the i^{th}-level decomposition, there are $m_{(i)}$ subsystems decomposed from each subsystem located on the upper level in Fig. 2, and each decomposed subsystem contains $N_{(i)}$ nodes with total $k_{(i)}$ links connecting these subsystems. The relationship between $N_{(i)}$ and $m_{(i)}$ are:

$$N_{(i-1)} = m_{(i)} N_{(i)} \tag{9}$$

$$N = m_{(1)} N_{(1)} = \ldots = \left(\prod_{1}^{i} m_{(i)} \right) N_{(i)} \tag{10}$$

The actual parallelism applied for different level depends on the parallel capabilities of the hardware platform, which greatly impacts the computational complexity. Therefore, the complexity analysis should be performed in two cases for each level: parallel case and sequential case.

Parallel Case: In this case, the matrix inversion processes for each block matrix are computed in parallel. If the $m_{(i)}$ decomposed block matrices after the $i^{(th)}$-level LDE decomposition can be computed in parallel, then the computational time for each subsystem in $(i-1)^{(th)}$-level that are decomposed into $m_{(i)}$ subsystems has the computational complexity of:

$$\begin{aligned} &O[f_p(N_{(i-1)}, k_{(i-1)})] \\ &= O[f(N_{(i)}, k_{(i)}) + {k_{(i)}}^3 + t_{p(i)}], i < r \end{aligned} \tag{11}$$

$$O[f_p(N_{(r-1)}, k_{(r-1)})] = O[N_{(r)}^3 + {k_{(r)}}^3 + t_{p(r)}], i = r \tag{12}$$

where $f(N_{(i)}, k_{(i)})$ is the computational complexity for the $(i)^{th}$-level decomposed subsystems: $f = f_p$ if the $(i)^{th}$-level can also be computed in parallel, else $f = f_s$. $t_{p(i)}$ denotes the overhead of launching the i^{th} level threads for parallel computation, which cannot be neglected and can even be the dominant part of the overall cost when i is large. Because generally the performance will slow down for the higher level parallel computation in the nested parallelism of the compute platforms such as the GPU.

Sequential Case: In this case, the the matrix inversion processes can only be computed in sequential, which makes the computational time for each subsystem in $(i-1)^{(th)}$-level after the $i^{(th)}$-level LDE decomposition has the computational complexity of:

$$O[f_s(N_{(i-1)}, k_{(i-1)})] = O[m_{(i)} f_s(N_{(i)}, k_{(i)}) + {k_{(i)}}^3] \tag{13}$$

$$O[f_s(N_{(r-1)}, k_{(r-1)})] = O[m_{(r)} N_{(r)}^3 + {k_{(r)}}^3], i = r \tag{14}$$

Here, the inversion of $m_{(i)}$ decomposed block matrices is computed in sequential, and in this work, if the $(i-1)^{(th)}$-level could not be parallelized, then the $i^{(th)}$-level could only be computed in sequential. Then the total complexity $O[f(N_{(0)}, k_{(0)})]$ actually can be obtained in a recursive way, as illustrated in Fig. 3, assuming the $(1 \sim q-1)^{th}$ level computation is parallelized and $(q \sim r)^{th}$ level computation is sequential.

C. SPECIFIC DECOMPOSITION PRINCIPLES

Based on the complexity analysis, the number of decomposition levels and the number of decomposed subsystems in each level can be evaluated given a specific power system topology and parallel platform. Generally, two decomposition principles are proposed to improve the decomposition performance.

Principle 1: In the $(i = 1 \sim r-1)^{th}$ level, make the number of connecting links $k_{(i)}$ to be smaller than the size of the decomposed subsystems $N_{(i)}$; and in the r^{th} level, make a balance between $k_{(r)}$ and $N_{(r)}$.

This principle is inspired by (11)-(14), as can be seen that if $k_{(i)} > N_{(i)}$ in the $(i = 1 \sim r-1)^{th}$ level, then $O[k_{(i)}^3]$ will be the dominant part of the complexity no matter in the parallel case or sequential case, which means, there is no need to perform a higher level decomposition. And in the r^{th} level, the balance should be made between $k_{(r)}$ and $N_{(r)}$ because it is

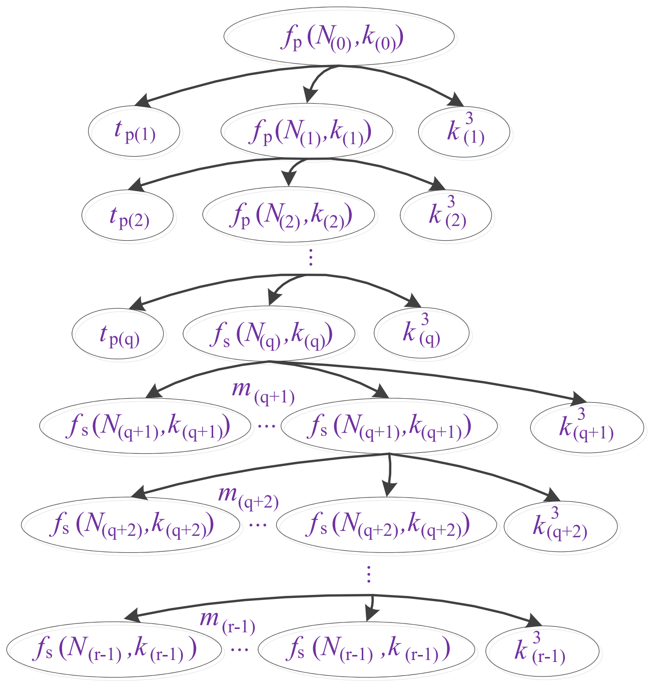


FIGURE 3. Recursive complexity analysis of the hierarchical LDE decomposition.

the last level and should make sure that $O[f(N_{(r-1)}, k_{(r-1)})]$ is minimized, just like the one-level traditional LDE method.

Principle 2: The launching of a higher level should achieve a lower computation time but not result in a larger latency, that is, for parallel computing:

$$O[f(N_{(i)}, k_{(i)}) + k_{(i)}{}^3 + t_{p(i)}] < O[N^3_{(i-1)}] \tag{15}$$

where $t_{p(i)}$ contains the overhead of launching child kernels for parallel computation as well as the latency of synchronization between the kernels. And for sequential computing:

$$O[m_{(i)} f_s(N_{(i)}, k_{(i)}) + k_{(i)}{}^3] < O[N^3_{(i-1)}] \tag{16}$$

This principle is used to judge whether a deeper decomposition is required, because the common parallel platforms such as GPU have limited capabilities of parallelism and the overhead of launching child kernels is significant. For sequential computation, the decomposition should also make sure the sum of computational time for each subsystem computation is smaller than the latency without decomposition. For example, when decomposing a 60 × 60 block matrix into 3 20 × 20 block matrices for sequential computing, it should be guaranteed that $3 \times t_{20} + t_{k20} < t_{60}$, where t_{20} denotes the computational time of inverting a 20 × 20 block matrix, and t_{k20} denotes the computing time for the **Q** matrix with size of $k \times k$ and correcting the block matrices with the **Q** matrix. Practically, the computational time of inverting a matrix given a specific size can be evaluated in advance, then the decision can be made on whether a deeper decomposition is necessary or not.

IV. CPU-BASED SEQUENTIAL AND GPU-BASED PARALLEL IMPLEMENTATION

The dynamic parallelism feature [18] of GPUs enables nested kernel function execution, which is suitable for the H-LDE decomposition architecture. In this section, both the CPU-based sequential and GPU-based parallel implementation of the IEEE 118-bus [19] test system is described for demonstration.

A. SEQUENTIAL AND PARALLEL CONFIGURATION

The IEEE 118-bus power system [19] is chosen as the test system to show the application of the proposed H-LDE method, which contains 118 buses, 54 generators, 177 lines, 9 transformers, and 91 loads. The equivalent network topology is illustrated in Fig. 4, where the bus number is shown on each node. For sequential H-LDE computation, let $r = 4$, $q = 0$, which means there are 4 level of LDE decomposition applied. The reason of choosing $r = 4$ is explained in the following part, and when the system scale increases, the level of LDE decomposition can increase to achieve the optimal performance.

For parallel implementation, the dynamic parallelism feature [18] of GPUs is utilized. The dynamic parallelism enables the kernel function to create new kernel functions on the GPU device dynamically. For example, the grid A that is a collection of several parallel threads is the first-level parallelism, in which every thread launches a new grid B. Grid A is called a "parent" grid, and the one launched by it is called "child" grid. Launching a set of new "child" grids also introduces a considerable cost including the latency of launching kernels and synchronizing these kernels. Therefore, if the child kernels do not extract much parallelism and there is not much benefit against their non-parallel counterparts, then the little benefit may be canceled out by the child kernel launching overheads. As an example, we use $r = 4$, $q = 2$ for parallel H-LDE computation of the 118-bus power system, which means there are 4 levels of LDE decomposition in total and the first two levels are computed using the GPU dynamic parallelism, while the 3^{rd} and 4^{th} level are computed sequentially.

B. TEST SYSTEM DECOMPOSITION

Following the two principles proposed in Section III(C), the 4-level H-LDE decomposition is shown in Fig. 4. The partition lines are highlighted in different line types for different levels, and the number shown besides a partition line denotes the number of links connecting the decomposed subsystems, that is, $k_{(i)}$. Note that this work only shows a specific partition, which may not be the optimal solution.

First-Level: As shown in Fig. 4, in the 1^{st} level decomposition, the 118-bus is decomposed into 4 subsystems: SS-1, SS-2, SS-3 and SS-4 with sizes of 30, 30, 28 and 30 respectively. The number of links connecting the decomposed subsystems is 15, which means the size of **Q** matrix is $k_{(1)} = 15$. This decomposition follows the *Principle 1*, making $k_{(1)} < N_{(1)}$, then after the inversion of the 30 × 30 and 28 × 28 matrices is computed in sequential or parallel, $\mathbf{G}_d^{-1}$ is obtained, and finally the improved LDE solver procedure (7) can be applied to

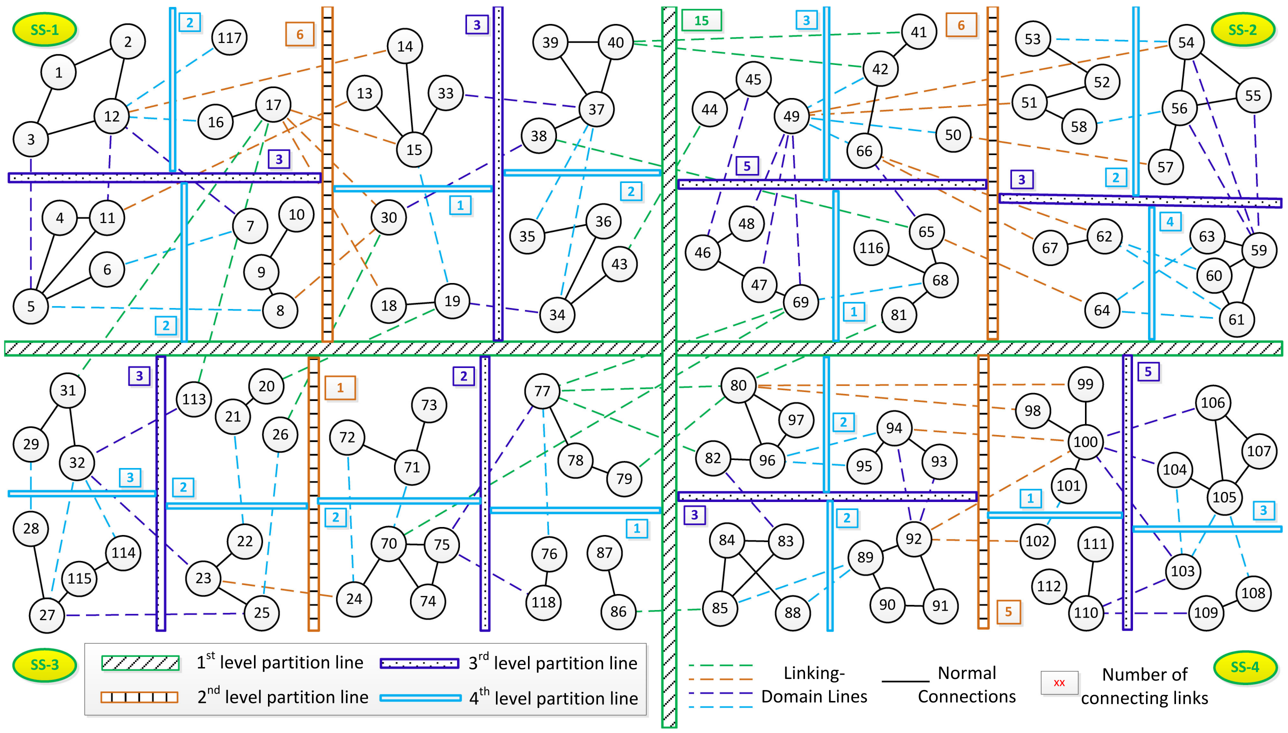


FIGURE 4. Topology partitioning of the IEEE 118-Bus test power system using the 4-level LDE decomposition.

solve the unknown state variables, which is extremely simple and can achieve a good speed-up.

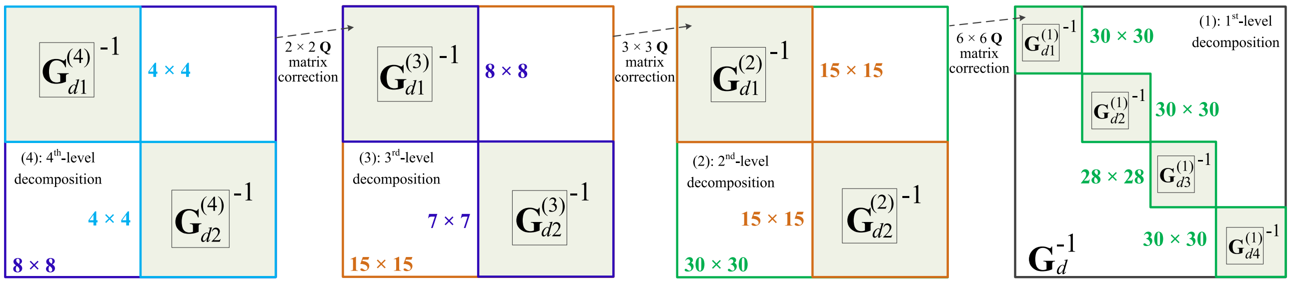
Second-Level: The 2nd $\sim$ 4th-level decomposition is applied to compute the inversion of the SS-1, SS-2, SS-3 and SS-4 conductance matrices, therefore, the actual computation sequence is performed as a bottom-up pattern, from the 4th-level to the 2nd-level. Taking SS-1 as an example, the 30 nodes are decomposed into 2 subsystems with the block matrix size of 15×15. This is also decomposed following *Principle 1*, making $k_{(2)} < N_{(2)}$. For parallel computing, as instructed by *Principle 2*, the overhead of launching the second-level parallelism should be smaller than the benefit from parallel computation of the block matrices inversions. We can see that $k_{(2)} = 6$ and $N_{(2)} = 15$, which could meet the requirement (15) through experimental results; that means the second-level decomposition could benefit from the parallel computation using a small linking-domain matrix. For sequential computing, the second-level decomposition could obviously achieve better speed-ups since computing the 30×30 matrix inversion involves much more computational efforts than computing two 15×15 matrices inversions. The second-level decomposition for SS-2, SS-3 and SS-4 has the same logic and procedure.

Third-Level: The third level block matrix inversion is computed in sequential as configured in Section IV(A), which means that the decomposition should take the actual computing time of the matrix inversion with a specific size into consideration. For example, the 15×15 block matrix is decomposed into 2 block matrices with sizes of 8×8 and 7×7, and based on *Principle 2*, the decomposition should satisfy $t_8 + t_7 + t_{k3} < t_{15}$, where t_x denotes the computational time if inverting a $x \times x$ block matrix, and t_{k3} denotes the computational time of inverting the generated $\mathbf{Q}$ matrix and correct the block matrix with the $\mathbf{Q}$ matrix. Since in this partition $k_{(3)} = 5$ for the worst case, it can be verified on the implemented computing platform (both CPU and GPU) that the requirements can be satisfied.

Fourth-Level: The final level decomposition follows the same logic as the third level decomposition, but as indicated in *Principle 1*, it should also make the balance between $k_{(4)}$ and $N_{(4)}$. In fact, this principle is not very rigorous, and in this case, $N_{(4)} = 4$ or 3, $k_{(4)}$ is also equal to or smaller than 3.

Fifth or Higher Level: From Fig. 4 it can be seen that for $N_{(4)} = 4$ or 3, making a higher level LDE decomposition is not necessary. Because computing the inverse of a 4×4 matrix does not involve much overhead, and if it is decomposed into smaller matrices, computing smaller matrix inversions and correcting them with the $\mathbf{Q}$ matrix will introduce extra latencies, which are larger than the benefit of decomposition and simply violate the *Principle 2*.

From the above multi-level decomposition configurations, it can be observed that the maximum matrix that is actually required to be inverted is 4×4 for block matrices and 6×6 for $\mathbf{Q}$ matrices ($k_{(2)} = 6$). The 15×15 $\mathbf{Q}$ matrix ($k_{(1)} = 15$) is not required to be inverted due to the proposed improved LDE computation procedure (7); the 30×30 and 28×28 block matrices inversions are actually assembled using the inverted 4×4 and 3×3 block matrices of the 4th-level



$\mathbf{G}_{d1}^{(4)\,-1}$
4 × 4
(4): 4th-level decomposition
4 × 4
$\mathbf{G}_{d2}^{(4)\,-1}$
8 × 8
2 × 2 Q matrix correction
$\mathbf{G}_{d1}^{(3)\,-1}$
8 × 8
(3): 3rd-level decomposition
7 × 7
$\mathbf{G}_{d2}^{(3)\,-1}$
15 × 15
3 × 3 Q matrix correction
$\mathbf{G}_{d1}^{(2)\,-1}$
15 × 15
(2): 2nd-level decomposition
15 × 15
$\mathbf{G}_{d2}^{(2)\,-1}$
30 × 30
6 × 6 Q matrix correction
$\mathbf{G}_{d1}^{(1)\,-1}$
30 × 30
(1): 1st-level decomposition
$\mathbf{G}_{d2}^{(1)\,-1}$
30 × 30
28 × 28
$\mathbf{G}_{d3}^{(1)\,-1}$
30 × 30
$\mathbf{G}_{d4}^{(1)\,-1}$
$\mathbf{G}_{d}^{-1}$

FIGURE 5. **Assembling process for inverting the block matrices of the first-level decomposition with a 4-level H-LDE decomposition.**

decomposition, as shown in Fig. 5. Therefore, the computational effort of H-LDE is greatly reduced compared to the original LDE method that computes the 30 × 30 and 28 × 28 block matrices inversions directly.

C. AUTOMATIC SYSTEM DECOMPOSITION

In this example, the test power system is decomposed manually to show the detailed application of the proposed H-LDE method. For the automatic decomposition of a specific level, reducing the number of links between sub-blocks is the key point to reduce the computation time. In fact, that is also an important task in sparse solvers and this problem of optimal decomposition is essentially the k-section problem in graph theory. The bisection and k-way heuristic algorithms are widely used to solve this kind of problem, and the related packages are also available in Metis [20], KaHIP [21], Scotch [22], etc. However, there is still lot of work to do to decompose a power system at multi-level, since the number of decomposition levels is variable. If m_i is a constant, then the k-section algorithm can be used in each level of decomposition until the minimum node set is reached, which can be determined in advance according to the experimental computational latency on the used computation platform. For example, in this case, the minimum size of the node set is set at 3. Integrating the proper partitioning algorithms into the multi-level decomposition of H-LDE method remains to be investigated in the future work.

V. SIMULATION RESULTS AND VERIFICATION

In this section, the matrix equations of the IEEE 39, 57, 118, and 300 bus benchmark test power systems [23] and the auto-generated 400/500/600-bus power systems are solved using the H-LDE method, and the speed-ups are evaluated on both the Intel$^{\text{TM}}$ i5-7300HQ 2.GHZ CPU with 8 G RAM and the NVIDIA$^{\text{TM}}$ Tesla V100 GPU platform with 5012 cores [24] by comparing with the Gauss-Jordan method, original LDE method, LU factorization with Gauss's algorithm [4] and KLU sparse matrix equation solution [14] method. The synchronous machine and other equipment models used in the test cases are the same as those of PSCAD/EMTDC$^{\text{TM}}$ [13]. The AC4A type exciter control is also attached to the machine model.

A. SPEED-UP OF GPU-BASED PARALLEL H-LDE COMPUTATION

The performance evaluation of GPU-based parallel H-LDE computation is separated from the CPU-based sequential computation, because they have different orders of magnitude in latencies and different application contexts. In this case, the conductance is regarded as changeable during simulation, therefore, the Gauss-Jordan (GJ) method is chosen as the base, since the pure LU factorization without re-ordering has a slightly larger complexity for a changeable conductance matrix and could not expose much parallelism possibilities compared to the GJ method. The matrix equation solution time of the proposed H-LDE method is compared with the traditional Gauss-Jordan (GJ), Schur Complement (SC) and original LDE (O-LDE) method. In this work, 2-level dynamic parallelism is exploited: the computation procedure of the GJ method could be not expose much parallelism, although some rows and columns of the pivoting or reduction operations can be computed in parallel; the matrix inversion of the block matrices generated by the SC and O-LDE method could be computed in parallel for the first level parallelism, but the application of the second level parallelism for computing the decomposed block matrices in parallel should be evaluated for different sizes of the block matrices due to the overhead of launching child kernels. For the H-LDE method, the 4-level decomposition and 2-level dynamic parallelism are already described in Section IV(B). Note that since the history item updating also occupies a considerable time for each power equipment, in this comparison, only the matrix equation solution time is recorded and compared.

The computational time under different number of decomposed subsystems for SC and O-LDE, and under different levels of H-LDE are shown in Fig. 6. Note that the GJ method was selected as the base, which is not shown in the figure. The time-step size is set at 20 μs, and 5000 steps of matrix equation solution time in total was recorded. From Fig. 6 we can see that the SC and O-LDE method can achieve their maximum speed-ups over the GJ method when m_{ss} reaches to 5 and 6 respectively. That is their full potential because they are both one-level decomposition methods. However, H-LDE could achieve the maximum speed-up of 36.1 over the GJ method at fourth level (4-L), nearly 2 times of performance over the original LDE method, which is quite significant.

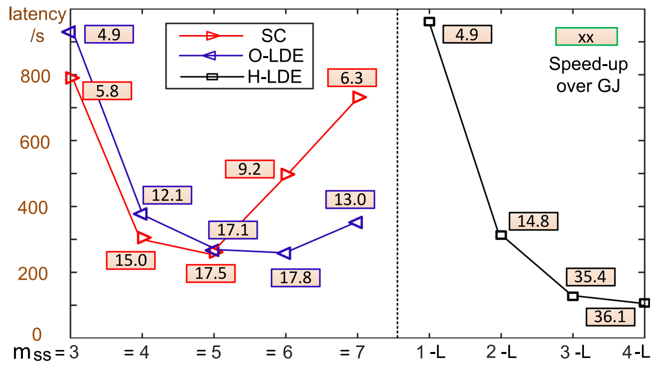


FIGURE 6. GPU-based computational time comparison between the SC, O-LDE and H-LDE method under different numbers of decomposed subsystems and different decomposition levels (Latencies of 5000 time-steps of matrix equation solution).

Besides, it can also be observed that as the number of levels increases, the improvement of speed-up slows down, which means that the hierarchical LDE could only divide the topology with a certain number of levels to achieve the maximum performance, and when r is greater than that number, the performance will slow down, as analyzed in Section IV(B).

B. SPEED-UP OF CPU-BASED SEQUENTIAL H-LDE COMPUTATION

The sequential computation is commonly used in EMT power system simulators, and in this case, the IEEE 39, 57, 118, and 300-bus benchmark test power systems are evaluated using the H-LDE method. To extend the system scale, the 400, 500 and 600-bus topologies are also generated using the randomized link generation with the row density of 4, which is in the typical row density range of power system conductance matrices [14]. Typically, there are two types of circuits that may influence the selection of proper solvers: circuit with constant conductance matrix, such as the IEEE 118-bus system; circuit with changeable conductance matrix, such as the AC power system with switches installed or the multilevel modular converter (MMC) circuit in AC/DC grids. All of the IEEE benchmark AC test power systems have constant conductance matrices, in this work, to obtain a changeable conductance matrix, several time-varying loads are installed in the power system. For example, in the IEEE 118-bus power system, the original consumed active and reactive power of the load on Bus 3 are 0.414 pu and 0.1062 pu; in this case study, the consumed power is changing from [0.8-1.2] times of the original load every 1 ms, that is, every 50 time-steps with the 20 μs time-step size.

Constant Conductance Matrix. For a constant conductance matrix, the LU factorization and KLU sparse solver have obvious advantages over the GJ method, since the L and U matrices can be computed in advance, and in the subsequent time slots solving $\mathbf{LU}x = b$ can be simplified into the forward and backward substitution: solving $\mathbf{L}y = b$ and $\mathbf{U}x = y$. Similarly, the corresponding matrices ($\mathbf{G}_d^{-1}$, $\mathbf{C}$, and $\mathbf{Q} = (\Lambda^{-1} + \mathbf{C}^T\mathbf{G}_d^{-1}\mathbf{C})^{-1}$) in the H-LDE method (6) can also

TABLE I Computational Time and Speed-Ups of 5000 Steps With Constant Matrix

Scale	LU	KLU	H-LDE	Sp-LU	Sp-KLU
39-bus	12 ms	12 ms	4 ms	3.00	3.00
57-bus	24 ms	23 ms	9 ms	2.67	2.56
118-bus	94 ms	59 ms	26 ms	3.62	2.27
300-bus	591 ms	161 ms	150 ms	3.94	1.07
400-bus	1,024 ms	249 ms	237 ms	4.32	1.05
500-bus	1,638 ms	298 ms	396 ms	4.14	0.75
600-bus	2,549 ms	361 ms	707 ms	3.61	0.51

[0]Sp-LU: H-LDE speed-up over LU factorization with Gauss's algorithm;
[0]Sp-KLU: H-LDE speed-up over KLU.

be obtained in advance. Note that in this case, the procedures (7)(8) are not required since they are targeting reducing the computational effort for a changing $\mathbf{Q}$ matrix but not reducing the storage cost. Therefore, for a constant conductance matrix, the H-LDE can be executed as:

$$\begin{aligned} v &= [\mathbf{G}_d^{-1} - \mathbf{G}_d^{-1}\mathbf{C}\mathbf{Q}\mathbf{C}^T\mathbf{G}_d^{-1}]i_{eq} \\ &= \mathbf{G}_d^{-1}i_{eq} - \mathbf{G}_d^{-1}\mathbf{C}\mathbf{Q}\mathbf{C}^T\mathbf{G}_d^{-1}i_{eq} \\ &= v_{DBM} - (\mathbf{G}_d^{-1}\mathbf{C})\mathbf{Q}(\mathbf{C}^T v_{DBM}) \end{aligned} \tag{17}$$

In this process, only the matrix multiplication operations are required, and since $\mathbf{G}_d^{-1}$ is a block diagonal matrix, and $\mathbf{C}$ only contains a small number of $1/-1$ elements with the rest of 0 elements, the computation of each time-step will be extremely fast. And compared to storing the entire matrix inversion, the storage cost is also reduced a lot, as analyzed in Section II(B).

The computational time (ms) of different test power systems are shown in Table I, the duration of simulation is 0.1 s, which is 5000 steps with 20 μs time-step size. Note that the latency is the pure matrix equation solution time without the power equipment circuit and history item updating latency involved for a pure comparison of different computational methods. In this case, the matrix input format for the KLU program is transferred into column compressed format in advance [14]. As can be observed in the results, the H-LDE method is always better than the LU factorization with Gauss's algorithm in the 600-node system scale, because without the re-ordering and pivoting techniques involved, the generated L+U matrix is a dense matrix and thus requires more computational effort compared to the simple multiplication operations in H-LDE.

However, since the sparse techniques are included in the KLU package, the H-LDE shows less scalability than KLU due to the sparsity of the generated L+U matrix. When the system scale is smaller 400-bus, H-LDE can achieve better performance; but when the system scale increases larger, the decomposed block matrix sizes increase, and then the influence of large storage and I/O cost could not be omitted, although the H-LDE method requires much less storage than the original LDE method. The scalability of H-LDE on different processors with different RAM sizes may be different, but this result at least shows that the H-LDE method can only achieve

TABLE II Computational Time and Speed-Ups of 5000 Steps With Changeable Matrix

Scale	LU	KLU	H-LDE	Sp-LU	Sp-KLU
39-bus	74 ms	138 ms	31 ms	2.39	4.45
57-bus	196 ms	339 ms	90 ms	2.18	3.77
118-bus	1,379 ms	953 ms	331 ms	4.17	2.88
300-bus	19,808 ms	2,536 ms	1,371 ms	14.45	1.85
400-bus	37,814 ms	3,493 ms	2,476 ms	15.27	1.41
500-bus	89,133 ms	4,538 ms	5,702 ms	15.63	0.80
600-bus	218,758 ms	5,101 ms	13,115 ms	16.68	0.39

better performance than the sparse LU method such as the KLU method within a certain system scale.

Changeable Conductance Matrix. For a changeable conductance matrix, the entire H-LDE computation procedure (6)(7)(8) is required; for example, the maximum size of matrix equation to be solved in the IEEE 118-bus configuration is 15×15 in solving (8), and the maximum size of matrix to be inverted is 6×6 as discussed in Section IV(B), which are much smaller than the 118×118 conductance matrix. Besides, if only the values of matrix elements change but the node connections do not change (that is, the non-zero elements locations do no change), the computation of the H-LDE method can also be accelerated a lot like the KLU method. The pre-processing of KLU including the permutation to block triangular form (BTF) and fill-reducing ordering could be reused for each time-step due to the same matrix pattern; and the H-LDE can hold the same multi-level partition at each time-step, which means the structure of assembling the high-level small inverted block matrices remains the same. Note that when nonlinear characteristics are inserted into the main circuit equation, for example, the surge arresters or nonlinear transformer models are applied, the solver needs to iteratively solve the equation. Therefore, the nonlinear case can also be regarded as the changeable conductance matrix case.

The computational time of 5000 time-steps of different test power systems are shown in Table II, the H-LDE speed-up over the LU factorization without sparse techniques involved is increasing as the system scale increases, which shows that the H-LDE method is more scalable than the pure LU factorization. For the KLU, since the pre-processing procedures are involved, it could not obtain good performance in the small scale system. But as the fill-in reducing algorithms are applied, the generated L+U matrix is still a sparse matrix and thus the method can scale very well when the system scale expands. For the H-LDE method, the computational procedure for computing the block matrices inversions can be significantly accelerated, and thus within the 400-bus system scale it can even achieve better performance than the sparse KLU method. It can be expected that if the CPU multi-core parallel architecture is utilized, the block matrices inversions can be computed in parallel to achieve a faster speed.

In EMT simulation, large power systems can usually be first decomposed into small subsystems using the latency-based transmission line models due to the small time-step size, and then the H-LDE method could be applied for the small subsystem simulation. Therefore, although sparse techniques are suitable for large-scale circuit simulation, the H-LDE method can also be applied for the fast and parallel EMT power system simulation within a certain power system scale.

VI. CONCLUSION

The linking-domain extraction (LDE) decomposition method is a new non-overlapping domain decomposition method that can be used for fast circuit simulation. However, the original LDE method is not efficient in computational procedure and not scalable due to the large storage cost. In this paper, the capability of LDE method is fully exploited by the proposed hierarchical LDE (H-LDE) method. First, the LDE computation procedure is improved to avoid storing the entire inverted conductance matrix to reduce the storage cost. Based on this, the detailed decomposition configuration for a specific number of decomposition levels is presented. Then, the complexity of H-LDE decomposition is analyzed, based on which the two decomposition principles are proposed to improve the decomposition performance. Based on the proposed decomposition principles, the IEEE 118-bus test power system is decomposed into 4 levels to demonstrate the application of the H-LDE method. The simulation results and speed-ups over the original LDE method, the classical LU factorization and the KLU sparse matrix equation solvers show that the proposed H-LDE method could achieve a better performance in the 400-bus level power system. Therefore, the H-LDE method can be applied for the fast and parallel power system circuit simulation within a certain power system scale. There is still room to improve the H-LDE method or explore the possibility of integrating some sparse techniques into the H-LDE method to fit for larger power system simulation problems in the future work.

REFERENCES

[1] L. O. Chua and P. Lin, *Computer-Aided Analysis of Electronic Circuits: Algorithms and Computational Techniques*. Englewood Cliffs, NJ, USA: Prentice-Hall, Inc., 1975.

[2] T. F. Chan and T. P. Mathew, "Domain decomposition algorithms," *Acta Numerica*, vol. 3, pp. 61–143, Jan. 1994.

[3] A. Toselli and O. B. Widlund, *Domain Decomposition Methods: Algorithms and Theory*. Berlin, Germany: Springer-Verlag, 2005

[4] F. N. Najm, *Circuit Simulation.* Hoboken, NJ, USA: John Wiley & Sons, 2010.

[5] K. Sun, Q. Zhou, K. Mohanram, and D. C. Sorensen, “Parallel domain decomposition for simulation of large-scale power grids,” *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des.*, San Jose, CA, USA, Nov. 2007, pp. 54–59.

[6] P. Aristidou, D. Fabozzi, and T. V. Cutsem, “Dynamic simulation of large-scale power systems using a parallel schur complement-based decomposition method,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 10, pp. 2561–2570, Oct. 2014.

[7] P. Aristidou, S. Lebeau, and T. V. Cutsem, “Power system dynamic simulations using a parallel two-level schur complement decomposition,” *IEEE Trans. Power Syst.*, vol. 31, no. 5, pp. 3984–3995, Sep. 2016.

[8] S. Y. R. Hui, K. K. Fung, and C. Christopoulos, “Decoupled simulation of multi-stage power electronic systems using transmission-line links,” in *Proc. Rec. IEEE Power Electron. Specialists Conf.*, Toledo, Spain, 1992, pp. 1324–1330.

[9] S. Y. R. Hui and K. K. Fung, "Fast decoupled simulation of large power electronic systems using new two-port companion link models," *IEEE Trans. Power Electron.*, vol. 12, no. 3, pp. 462–473, May 1997.

[10] J. E. Schutt-Aine, "Latency insertion method (LIM) for the fast transient simulation of large networks," *IEEE Trans. Circuits Syst. I*, vol. 48, no. 1, pp. 81–89, Jan. 2001.

[11] S. N. Lalgudi, M. Swaminathan, and Y. Kretchmer, "On-chip power-grid simulation using latency insertion method," *IEEE Trans. Circuits Syst. I*, vol. 55, no. 3, pp. 914–931, Apr. 2008.

[12] T. Duan and V. Dinavahi, "A novel linking-domain extraction decomposition method for parallel electromagnetic transient simulation of large-scale AC/DC networks," *IEEE Trans. Power Del.*, vol. 36, no. 2, pp. 957–965, Apr. 2021.

[13] *EMTDC^{TM} User's Guide: A Comprehensive Resource for EMTDC*, v4.6.0, Manitoba HVDC Research Centre, May 2018, Accessed: Dec. 15, 2019. [Online]. Available: https://www.pscad.com/uploads/knowledge_base/emtdc_manual_v4_6.pdf

[14] T. A. Davis and E. P. Natarajan, "Algorithm 907: KLU, a direct sparse solver for circuit simulation problems," *ACM Trans. Math. Softw.*, vol. 37, no. 6, pp. 36: 1–36:17, 2010.

[15] X. Chen, Y. Wang, and H. Yang, "NICSLU: An adaptive sparse matrix solver for parallel circuit simulation," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 32, no. 2, pp. 261–274, Feb. 2013.

[16] "SPICE: Simulation program with integrated circuit emphasis," EECS Dept., Univ. California at Berkeley, Accessed: Feb. 14, 2019. [Online]. Available: http://bwrcs.eecs.berkeley.edu/Classes/IcBook/SPICE/

[17] M. A. Woodbury, *Inverting Modified Matrices*. Princeton, NJ, USA: Princeton Univ. Press, 1950.

[18] *NVIDIA Corporation*, *Dynamic Parallelism in Cuda*, 2012, Accessed: Jan. 15, 2020. [Online]. Available: https://developer.download.nvidia.cn/assets/cuda/files/CUDADownloads/TechBrief_Dynamic_Parallelism_in_CUDA.pdf

[19] *PSCAD^{TM} IEEE 118 Bus System*, Revision 1, Manitoba HVDC Research Centre, Winnipeg, Manitoba, Canada, 2018, Accessed: Jan. 2, 2020. [Online]. Available: https://www.pscad.com/uploads/knowledge_base/ieee_118_bus_technical _note.pdf

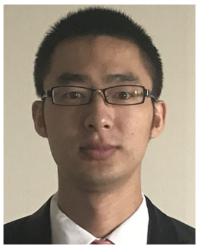
[20] G. Karypis and V. Kumar, "Multilevel k-way partitioning scheme for irregular graphs," *J. Parallel Distrib. Comput.*, vol. 48, no. 1, pp. 96–129, 1998.

[21] C. Schulz, "The graph partitioning framework KaHIP," *Online Project*, 2019, Accessed: Jan. 14, 2020. [Online]. Available: https://github.com/KaHIP/KaHIP

[22] *Scotch Project*, 2006. Accessed: Jan. 23, 2020. [Online]. Available: https://gforge.inria.fr/projects/scotch/

[23] 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.

[24] NVIDIA TeslaV100Datasheet. 2017, Accessed: Feb. 05, 2020. [Online]. Available: http://www.nvidia.com/content/PDF/Volta-Datasheet.pdf



TONG DUAN (Student Member, IEEE) received the B.Eng. degree in electronic engineering from Tsinghua University, Beijing, China, in 2013. He is currently working toward 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, domain decomposition, and parallel computing.

VENKATA DINAVAHI (Fellow, IEEE) received the B.Eng. degree in electrical engineering from the Visveswaraya National Institute of Technology, Nagpur, India, in 1993, the M.Tech. degree in electrical engineering from the Indian Institute of Technology 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.