GPU-Based Power Converter Transient Simulation with Matrix Exponential Integration and Memory Management

Wei Wu[a], Peng Li[a], Xiaopeng Fu[a] *, Zhiying Wang[a], Jianzhong Wu[b], Chengshan Wang[a]

[a] Key Laboratory of Smart Grid of Ministry of Education, Tianjin University, Tianjin 300072, China

[b] Institute of Energy, School of Engineering, Cardiff University, Cardiff CF24 3AA, UK

Abstract: With the extensive application of power electronic equipment in power systems, electromagnetic transient (EMT) simulation involving power converters becomes more challenging. Due to its multithreads and high throughput architecture, the graphics processing unit (GPU) can be used to accelerate those EMT simulations. A GPU-based matrix exponential method and its memory management for power converter transient simulation are proposed in this paper. A parallel exponential integration algorithm is established from two aspects to fully utilize the GPU multithreads capability. The matrix exponentials which are recomputed with the on/off state changes of power electronic switches are cached in GPU memory. The simulation efficiency is improved by reusing the cached data and reducing heterogeneous data transmission between CPU and GPU. Several strategies are experimented to manage the cache memory considering the EMT simulation workflow. The proposed memory management expands the simulation capability by substantially reducing the memory requirement and maintains the speed advantage of the GPU-based simulator. The proposed method is tested on wind power plants of different scales with power electronic interfaced wind generators. Simulation results indicate that the proposed method and its memory management expand the simulation capability and achieve speedups.

Key words: Electromagnetic transient simulation, power converter, graphics processing unit, matrix exponential integration, memory management.

* Corresponding author. Tel.: +86 158 2288 8351; fax: +86 22 27892810.
E-mail address: fuxiaopeng@tju.edu.cn.

1. Introduction

With the rapid development of renewable generation integration [1], power converters are widely used in power systems to condition nonstandard AC or DC electric power for the grids and the users [2]. The precise control of converters in the microsecond scale raises the need for high-fidelity electromagnetic transient (EMT) simulations [3]. Specifically, high-frequency switching devices within the converters require detailed PWM control model and small simulation time-steps to precisely locate switching events. To accurately study the transients of the power system, the converters are required to be simulated with their detailed model [4]. However, EMT simulation of these detailed modeling systems poses challenges toward simulators from both software and hardware perspectives. Switches in power systems lead to time-varying system matrices, which become a computational bottleneck during the repeated calculation. The simulation step size limitation arises when handling the switch operations, and it becomes very inefficient for large-scale systems with long simulation timespan.

In response to these challenges, a variety of research have been conducted. Network partition methods have been proposed for decoupling the time-varying portion from the main network equation. Among them is the conventional decoupling method through propagative transmission lines that provide natural delay [5], and the more recent Multiarea Thévénin Equivalent (MATE) framework [6] and the Nested Fast and Simultaneous Solution method [7], which have theoretical roots from Diakoptics. Another trend is to relax the stringent requirement on the step sizes by exploiting the timescale property of the power systems. Multirate method decouples the power system by latency and adopts different time-steps for subsystems [8]. Dynamic average-value models (AVMs) allows larger time-steps by simplifying the model of power converter equipment. AVMs are cost-effective when switching frequency harmonics are not of concern [9]. Dynamic phasor simulation results capture the envelopes of the power system waveforms, and the step size is not limited by the power frequency [10].

Exploiting the rapid advancements of parallel computing hardware is another major aspect to address the simulation challenges. It helps to resolve the conflict between the time-consuming calculation process and the necessary simulation precision of power systems with complicated controls. Instead of processor units with higher clock rates, the increased computation power is expected to be delivered through parallelism. Coarse-grained parallel EMT simulation on multi-core CPUs was reported for realistic large grids [11]. FPGAs (Field-Programmable Gate Arrays), which are intrinsic parallel hardware with pipelined architecture, are popular in the real-time simulation scenario [12]and are included in commercial real-time simulation platforms along with other processor units [13], [14]. In contrast to CPUs which are designed to optimize the performance of sequential codes with sophisticated control logic, GPUs are designed as parallel, throughput-oriented computing engines with single instruction, multiple thread (SIMT) execution model. In this model, multiple threads are processed by a single instruction that can be executed with different data parallel. Fine-grained parallel programs are suitable to be executed in GPUs and gain a significant acceleration. To date, this large performance gap between parallel and sequential execution has already motivated many applications to move the fine-grained parallel computing portions of the analysis program from CPU to GPU. For instance, GPU was used for sensitivity analysis of large power grid [15]. Research has also been devoted to exploring the potential parallelism of the EMT simulation algorithm. Professor Gole from the University of Manitoba first reported GPU-based EMT simulation in 2011 [16]. Under the nodal analysis framework, typical network components were simulated based on coarse-grained parallelism. It is particularly efficient for devices with many identical circuits, e.g., modular multi-level converters, where they were partitioned to separate massive submodules with identical structures from the arms [17]. Propagation delay-based decomposition was the frequently used tool to facilitate the coarse-grained parallelism [18]. To fully exploit the massive-threads parallel computing ability of GPUs, recent studies have also focused on the fine-grained parallelism. Notably, a cloud-computing based power system simulator, named CloudPSS [19] is developed featuring heterogeneous architecture and automatic code generation [20], [21].

In this paper, the exponential integrators (EIs) are adopted as the numerical integration method for the EMT simulator [22], which are capable of handling the stiffness property associated with power converter models [23]. EIs make use of the matrix exponential function and related functions to achieve good numerical stability and accuracy against its prototype integration formula, and allows larger simulation timestep [24]. This new integration method is L-stable, meaning that it's immune to the numerical oscillation, in additional to the A-stable property of the widely used trapezoidal method. It remains explicit and avoids the iterative calculation which provides fine-grained data parallelism for GPU-based simulator.

This paper proposes a GPU-based parallel exponential integration method for power converter transient simulation. Firstly, a parallel algorithm for EIs is established by fully exploring the parallel potential of EIs on GPU architecture. Then the recurring calculation of matrix exponential function is minimized by adopting a caching mechanism designing based on the memory hierarchy of the simulator. Additionally, the cache is dynamically managed by a dedicated strategy that suits for converter simulation, which is shown to enlarge the application range of the proposed method. Main contributions of this paper are:

a) A caching method for GPU-based parallel EIs, which alleviates the computation burden by fully exploring the repetitiveness of converter states. This method presents good speedup in time-varying system simulations, especially for power systems with many high-frequency power converters.

b) A memory management method, which updates cached data as the simulation progresses through different stages of the dynamics. It is based on several cache replacement strategies selected to fit the workflow of power converter transient simulation and achieves high hit rates and enhances the simulation of large scale systems.

The rest of this paper is organized as follows: Section II gives a brief summary of parallel EIs. Section III presents an efficient design of the GPU-based power converter transient simulation method. The mentioned memory management strategy is discussed in Section IV. In Section V, a detailed analysis of the overall performance of the GPU-based matrix exponential method and memory management strategy is presented. Finally, Section VI concludes the study.

2. Parallel exponential integrators

In the state variable-based exponential integration method, ordinary differential equations in the form of (1) are established to describe the dynamics of the studied electric power systems:

$$\begin{cases}\dot{\boldsymbol{x}} = \boldsymbol{f}(t, \boldsymbol{x}, \boldsymbol{u}) \\ \boldsymbol{y} = \boldsymbol{g}(t, \boldsymbol{x}, \boldsymbol{u})\end{cases} \tag{1}$$

where $\boldsymbol{x}$, $\boldsymbol{u}$ and $\boldsymbol{y}$ are the state variables, input and output variables, respectively. The state-space analysis for transient simulation can be illustrated, taking the small power system shown in Fig. 1 as an example. The transmission line is modeled as a PI-section of lumped parameter, and the P-Q loads are modeled as passive RLC elements. The inductor current I_{LT} of transformer primary circuit, the currents I_{L1}, I_{L2} of loads, the current I_{L3} of PI-section and capacitor voltages U_{C1}, U_{C2} of PI-section are selected as the state variables, as they suffice to determine all response of the system. The voltage source $\boldsymbol{U}_{\mathrm{S}}$ is regarded as the input variable and the standard state-space equations (1) are obtained by removing the algebraic variables and rearranging the basic equations of the circuit. Details of the prime circuit current of the transformer and the voltage of capacitors in PI-section are given in equations (2)-(3) below. And the standard state-space equations (4) are established by ensembling the equations from each state variable [25], where $\boldsymbol{A}_{\text{net}}$ and $\boldsymbol{B}_{\text{net}}$ are parameter matrices.

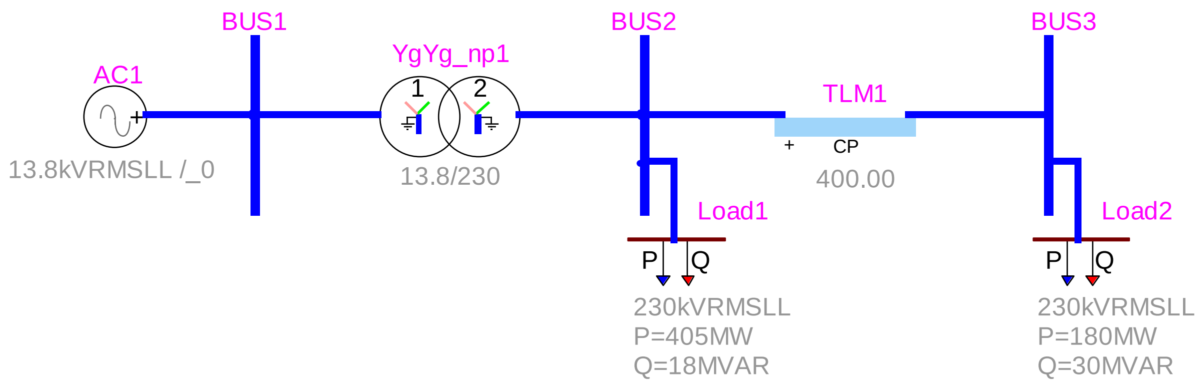

Fig. 1 A small power system example for state-space analysis.

$$\begin{bmatrix} \dot{I}_{\text{LTa}} \\ \dot{I}_{\text{LTb}} \\ \dot{I}_{\text{LTc}} \end{bmatrix} = \begin{bmatrix} -\frac{R_{1a}+R_{ma}}{L_{1a}} & 0 & 0 \\ 0 & -\frac{R_{1b}+R_{mb}}{L_{1b}} & 0 \\ 0 & 0 & -\frac{R_{1c}+R_{mc}}{L_{1c}} \end{bmatrix} \begin{bmatrix} I_{\text{LTa}} \\ I_{\text{LTb}} \\ I_{\text{LTc}} \end{bmatrix} + \begin{bmatrix} \frac{1}{L_{1a}} & -\frac{1}{L_{1a}} & 0 \\ 0 & \frac{1}{L_{1a}} & -\frac{1}{L_{1a}} \\ -\frac{1}{L_{1a}} & 0 & \frac{1}{L_{1a}} \end{bmatrix} \begin{bmatrix} U_{\text{sa}} \\ U_{\text{sb}} \\ U_{\text{sc}} \end{bmatrix}, \tag{2}$$

$$\begin{bmatrix} \dot{U}_{\text{C1a}} \\ \dot{U}_{\text{C1b}} \\ \dot{U}_{\text{C1c}} \end{bmatrix} = \begin{bmatrix} -\frac{1}{C_{1a}} & 0 & 0 & \frac{1}{C_{1a}} & 0 & 0 \\ 0 & -\frac{1}{C_{1b}} & 0 & 0 & \frac{1}{C_{1b}} & 0 \\ 0 & 0 & -\frac{1}{C_{1c}} & 0 & 0 & \frac{1}{C_{1c}} \end{bmatrix} \begin{bmatrix} I_{\text{L2a}} \\ I_{\text{L2b}} \\ I_{\text{L2c}} \\ I_{\text{L3a}} \\ I_{\text{L3b}} \\ I_{\text{L3c}} \end{bmatrix}, \tag{3}$$

$$\begin{bmatrix} \dot{\boldsymbol{I}}_{\text{LT}} \\ \dot{\boldsymbol{I}}_{\text{L1}} \\ \dot{\boldsymbol{I}}_{\text{L2}} \\ \dot{\boldsymbol{I}}_{\text{L3}} \\ \dot{\boldsymbol{U}}_{\text{C1}} \\ \dot{\boldsymbol{U}}_{\text{C2}} \end{bmatrix} = [\boldsymbol{A}_{\text{net}}] \begin{bmatrix} \boldsymbol{I}_{\text{LT}} \\ \boldsymbol{I}_{\text{L1}} \\ \boldsymbol{I}_{\text{L2}} \\ \boldsymbol{I}_{\text{L3}} \\ \boldsymbol{U}_{\text{C1}} \\ \boldsymbol{U}_{\text{C2}} \end{bmatrix} + [\boldsymbol{B}_{\text{net}}][\boldsymbol{U}_{\text{s}}] \tag{4}$$

The differential equations in (1) can be split into a linear part $\boldsymbol{A}$ and a nonlinear part $\boldsymbol{G}(t, \boldsymbol{x}, \boldsymbol{u})$:

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(t, \boldsymbol{x}, \boldsymbol{u}) = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{G}(t, \boldsymbol{x}, \boldsymbol{u}) \tag{5}$$

For any $t > t_0$, an analytical solution of (5) is available as:

$$\boldsymbol{x}(t) = \mathbf{e}^{(t-t_0)\boldsymbol{A}}\boldsymbol{x}_0 + \int_{t_0}^{t} \mathbf{e}^{(t-\tau)\boldsymbol{A}}\boldsymbol{G}\big(\tau, \boldsymbol{x}(\tau), \boldsymbol{u}(\tau)\big)d\tau \tag{6}$$

Motivated by the above semi-analytical formula, the EIs explicitly embed the matrix exponential function and related functions into the integration formula, collectively referred to as the φ functions. As a consequence of using exponential in this method, EIs all have L-stability as the traditional trapezoidal rule, which makes them suitable for solving stiff power system problems. Different types of explicit, implicit, single-step, and multi-step EI formulas are formed using φ functions of different orders, and those formulas provide better numerical stability and accuracy than their prototype integration formulas [22]. A variety of numerical integration methods are compared with EIs in [26]. The general class of EIs is expressed as a linear combination of φ function matrix-vector products. The computation of these matrix-vector products constitutes the major computation time of the EI-based EMT simulation.

Generally, there are two approaches for the solution of φ function-vector, which are the direct calculation approach and the Krylov subspace approximation approach [23], [24]. Direct calculation requires computing the matrix exponential function which is a process of $\mathrm{O}(n^3)$ complexity, while the Krylov subspace approach has a complexity roughly of $\mathrm{O}(nnz)$ [27]. Considering the expected acceleration of GPU, the direct calculation approach will be fast in small-to-medium scale cases, and the Krylov subspace method will be faster and more scalable for larger-scale cases. An illustration of the application range of the two approaches is given in Fig. 2, where the dimension of the studied system is on x-axis and the computation time is on y-axis. When the size of the studied system is smaller than N_{d0}, direct calculation (black) outperforms the Krylov subspace method (blue), and vice versa. Other methods Fig. 2 in drew with other colors will be explained when they are introduced in the following sections.

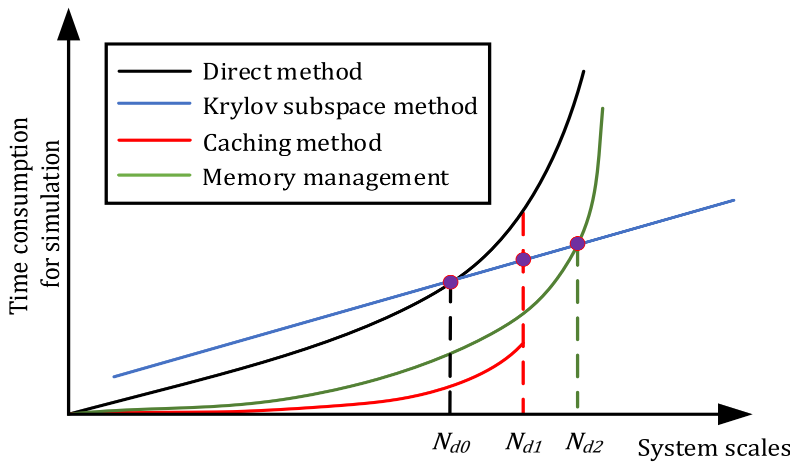

Fig. 2 Illustrative diagram for simulation time comparison of different EI solution methods: direct calculation method, Krylov subspace method, memory management method and caching method, in different simulation scales.

For the implementation of EIs, the state variable $\boldsymbol{x}$ is updated by a series of matrix-vector multiplications and vector additions, with the calculations of matrix exponential functions being the most computationally intensive. These operations are naturally fine-grained, and they conform to the single instruction multiple threads (SIMT) execution model to exploit the massive threads on GPU. Hence, by offloading those operations from CPU to GPU, the simulation performance will gain. On account of the stream technique in GPU programming, data transmissions can be overlapped with the kernel functions called by GPU. The parallel process of exponential integration methods is established on these two aspects.

3. Power converter transient simulation method based on GPU

3.1. GPU Memory Hierarchy

In typical EMT simulation workflows, there are parallel throughput-oriented computing tasks that can achieve higher performance on GPUs, and serial and few-threads operations which performs better on CPUs with lower operation latencies. Therefore, the joint CPU-GPU execution is required for optimal EMT simulation performance. This paper achieves parallel EMT simulation with the compute unified device architecture (CUDA) platform [28].

As shown in Fig. 3, a CUDA device supports several types of memories [29]. In the state-of-the-art GPUs, there is dynamic random access memory (DRAM) called global memory. It can be read and written by the GPU codes, and the kernel functions need global memory to store pertinent data transferred from the main memory. There are shared memory and registers that are on-chip memories. Registers are thread-private, meaning that each thread can only access its own register. Shared memory is block-private, all threads in a block share the data in shared memory.

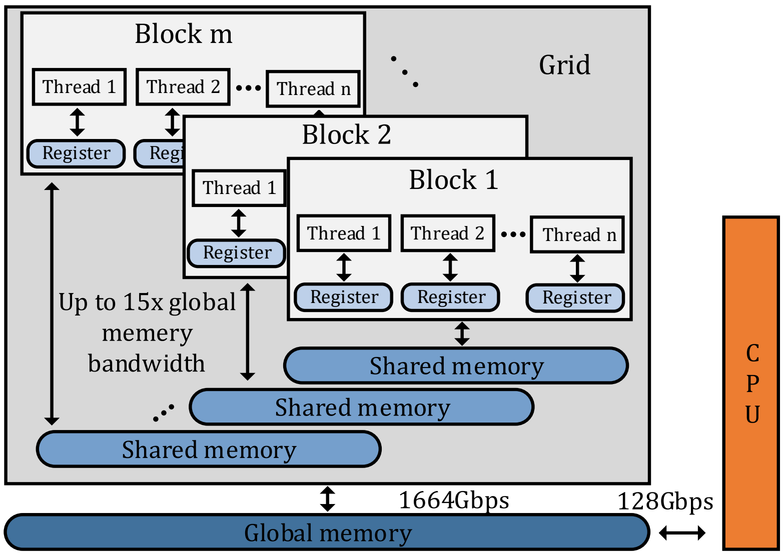

Fig. 3 CUDA memory hierarchy model.

Since traditionally GPUs need to achieve high pixel fill rate, their global memory is able to provide much higher bandwidth than the CPU memory. As shown in Fig. 3, the global memory bandwidth of NVIDIA GPU used in this paper can achieve 1664Gbps when heterogeneous data transmission speed through PCI express 2.0 x16 bus is only 128Gbps. This huge bandwidth gap motivates careful management of the precious GPU global memory to store only the most pertinent values. Variables in shared memory and register can be accessed at very high speed in a highly parallel manner; however, the amount of shared memory per block is too small to save large dimension matrices during realistic-size system simulations.

3.2. Implementation of GPU-based Power Converter Transient Simulation Method

The EMT simulation of the system consisting of a large volume of power converters is quite different from traditional EMT simulation. For the exponential Euler formula [22]:

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n + h\boldsymbol{\varphi}(h\boldsymbol{A})\big(\boldsymbol{A}\boldsymbol{x}_n + \boldsymbol{G}(t_n, \boldsymbol{x}_n, \boldsymbol{u}_n)\big), \tag{7}$$

matrix $\boldsymbol{A}$ changes with the states of power electronic switches constantly, and the φ function matrix $\boldsymbol{\varphi}(h\boldsymbol{A})$ changes along with $\boldsymbol{A}$. The formula is essential:

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n + h\boldsymbol{\varphi}\left(\sum_{i=1}^{p} hA_iS_i\right)\left(\sum_{i=1}^{p} A_iS_i\boldsymbol{x}_n + \boldsymbol{G}(t_n, \boldsymbol{x}_n, \boldsymbol{u}_n)\right), \tag{8}$$

where p is the status combination number of power electronic switches; $S_i (i = 1, \dots, p)$ are the binary switching functions that present the on/off status of the corresponding switches; $A_i (i = 1, \dots, p)$ are the coefficient matrices depending on the topology and parameters of the converters. Furthermore, the action time of the switches is located by interpolation. The power converter EMT simulation follows these steps:

a) Solve the electrical system/control system by parallel EIs, query the switches' status change, go to d) if no switch operations are returned;

b) Detect the switch action time and update the states at this moment by interpolation, update the state matrix;

c) Update the $\boldsymbol{\varphi}$ function matrix, step forward, then return to the time mesh through back-interpolation;

d) Perform the simulation in the next time-step by parallel EIs on GPU.

In the GPU-based implementation of exponential integration, φ functions are calculated using a rational approximation in CUDA. Intermediate vectors and matrices are stored in the GPU memory to save the delay in data transmission between CPU and GPU. To facilitate heterogeneous data transmission, CUDA streams are created to manage the concurrency of the programs.

As presented in Fig. 4, the calculation process of φ functions is performed in GPU with streams. In the rational approximation of the φ functions, a solution kernel function is designed which uses LU factorization. Operations in different streams are performed concurrently. When no stream is assigned to an operation, the default stream is set and the operation in default stream must wait until the previous operation has completed. There are two memory copy operations from CPU to GPU and one from GPU back to CPU and one solution kernel function operation in default stream showed in the upper half of Fig. 4. To illustrate the process of streams, it is assumed a copy operation time is t_c and a solution kernel function operation time is t_k, because all matrices involved have the same dimension. $\boldsymbol{A}$ and $\boldsymbol{B}$ represent the corresponding matrix polynomials. If $\boldsymbol{B}$ is divided into two column blocks $\boldsymbol{B}_1$ and $\boldsymbol{B}_2$, which is assigned to two streams, the kernel function tasks, and copy tasks can be overlapped with each other. As a result, nearly one copy operation time and half solution kernel function operation time $t_k/2 + t_c$ is saved.

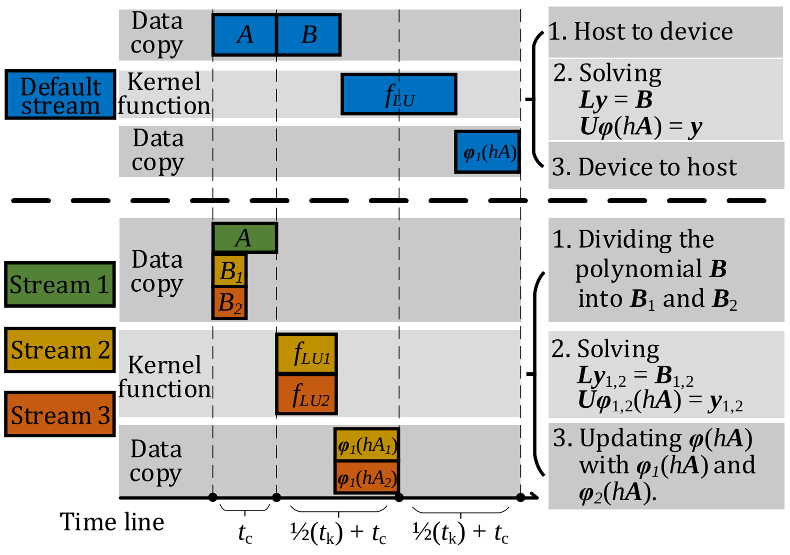
Default stream
Data copy
A
B
Kernel function
f_{LU}
Data copy
$\varphi_1(hA)$
1. Host to device
2. Solving
$\boldsymbol{Ly} = \boldsymbol{B}$
$\boldsymbol{U}\varphi(h\boldsymbol{A}) = \boldsymbol{y}$
3. Device to host
Stream 1
Stream 2
Stream 3
Data copy
A
B_1
B_2
Kernel function
f_{LU1}
f_{LU2}
Data copy
$\varphi_1(hA_1)$
$\varphi_1(hA_2)$
1. Dividing the polynomial $\boldsymbol{B}$ into $\boldsymbol{B}_1$ and $\boldsymbol{B}_2$
2. Solving
$\boldsymbol{Ly}_{1,2} = \boldsymbol{B}_{1,2}$
$\boldsymbol{U}\varphi_{1,2}(h\boldsymbol{A}) = \boldsymbol{y}_{1,2}$
3. Updating $\varphi(h\boldsymbol{A})$ with $\varphi_1(h\boldsymbol{A})$ and $\varphi_2(h\boldsymbol{A})$.
Time line
t_c
$½(t_k) + t_c$
$½(t_k) + t_c$

Fig. 4 Solving process of φ function using CUDA streams.

3.3. Caching Data in GPU Global Memory

For power systems containing large-scale power converters, the matrix $\boldsymbol{A}$ and φ functions require updating when switch on/off states or circuit topology changes, due to the high-frequency characteristic of power electronic switches, the solution of φ functions is the most time-consuming part, for the case study that will be presented in Section V, the solution of φ functions accounts for 69% of the total time for the entire simulation. It is worth noting that, although the state of the power electronic switches changes at a high frequency, the power system is periodic in nature, switching states may appear multiple times during the simulation. Therefore, φ functions calculated based on the state matrices are cached, which can be retrieved when the corresponding switch state appears again. In this way, a lot of repeating calculations are saved.

The caching method is designed based on memory hierarchy of the simulator in this paper. In the studied problem of this paper, data transmissions between CPU and GPU are much slower than GPU memories, and the φ functions are calculated frequently, which motivates the cache procedure to be performed in GPU. Since φ functions are system-scales matrices, they need a relatively large space to be cached, then the global memory in GPU is chosen for the balance between speed and memory size.

The flowchart of Fig. 5 gives the process of caching method for power converter transient simulation. Since the state matrices are formed with the on/off states of switches, the control system is solved firstly to derive the switch states. The electrical system is then solved if the switch combination is cached; otherwise, it is necessary to generate a state matrix and calculate φ function. Once the φ function is calculated, it is cached in GPU global memory.

The impact of caching φ function matrices is made clear by adding to the illustration diagram presented in Section II. Caching φ functions in GPU global memory, as drawn in Fig. 2, the red curve shows a significant acceleration compared with the direct calculation method. Moreover, caching method broadens the range of application of direct calculation method from the scale of N_{d0} to N_{d1}. Nevertheless, due to the limited GPU global memory, caching all φ functions fails at the scale of N_{d1} and it degenerates to the direct calculation method.

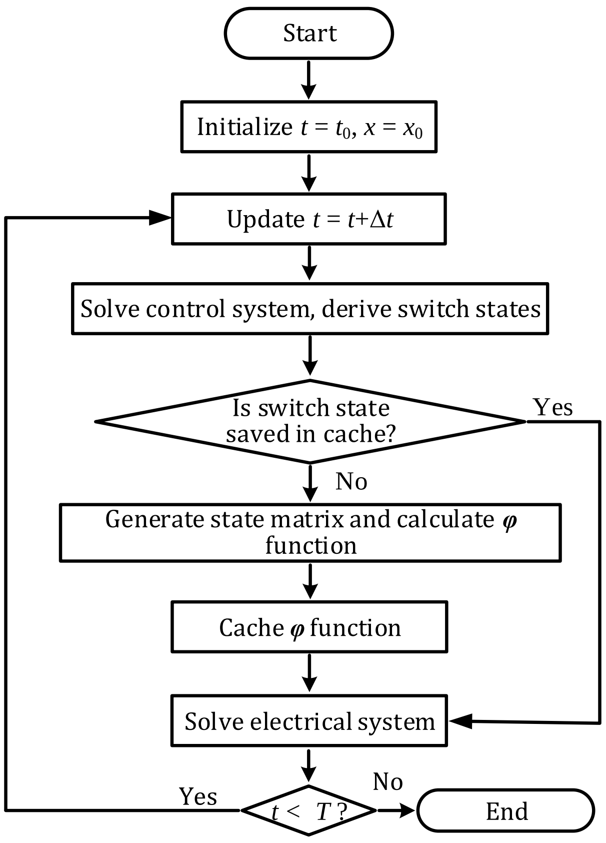
Start
Initialize $t = t_0$, $x = x_0$
Update $t = t+\Delta t$
Solve control system, derive switch states
Is switch state saved in cache?
Yes
No
Generate state matrix and calculate φ function
Cache φ function
Solve electrical system
Yes
$t < T$?
No
End

Fig. 5 Flowchart of the caching method for power converter transient simulation.

4. GPU memory management

When performing GPU-based EMT simulation with the caching method, it must be considered that it cannot be assumed that the GPU memory can hold all the required computation matrices, especially for large-scale cases or when many power converters are present. When the storage requirement is beyond the cache size, data chosen according to a particular principle is replaced out of the cache.

4.1. GPU Memory Management for Power Converter Simulation

The difficulties in data storage are twofold. Since power converters utilize many switches, the number of the switch states grows exponentially as well as the number of φ functions. The φ functions matrices are also of high dimension growing with the system dimension. Due to these difficulties, the φ functions cannot always all fit into the GPU global memory. A GPU memory management gives a solution to manage the limited global memory to perform the large-scale power converter transient simulation. The implementation of φ functions caching method and the memory management in the EMT simulation is presented in Fig. 6.

GPU has limited global memory, allocating excessive memory space as a cache, GPU may encounter problems that lead to program interruption; on the other hand, allocating insufficient cache space will cause frequent data replacement, which leads to repeated φ function calculations and increased computation time. Therefore, the size of the pre-allocated cache memory affects the speed of the program. To get expected program speed, determining a proper cache size is necessary.

The addition of the memory management layer on top of the caching method resolves the failure of the caching method previously mentioned. As shown in Fig. 2, the proposed method presented by the green curve broadens the application range of direct calculation method further from the scale of N_{d1} to N_{d2}, although a small performance penalty is expected in the implementation.

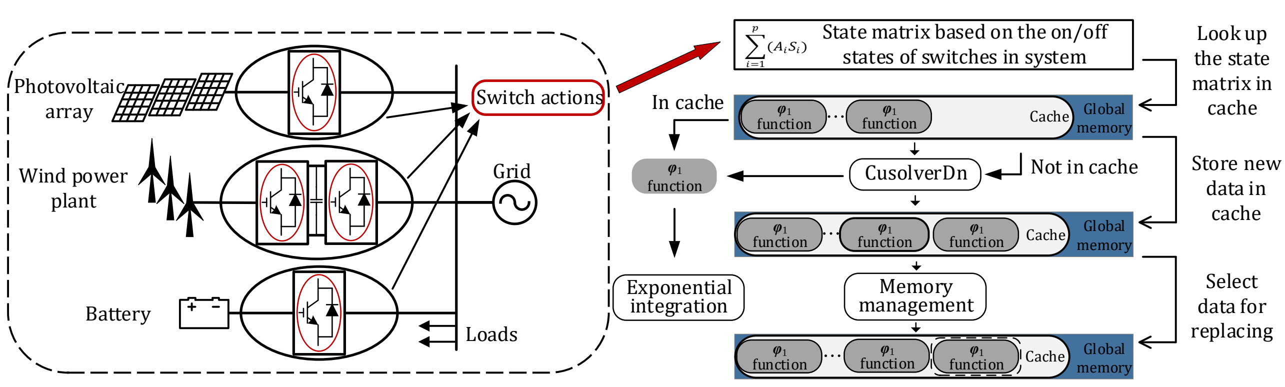

Fig. 6 Implementation of φ functions caching method and its memory management in the EMT simulation of power system containing large-scale power converters.

4.2. Caching Replacement Strategies-based GPU Memory management

For the power converter simulation with many power electronic switches, the proposed method manages the φ functions storage in GPU global memory based on the recency, frequency and historical information of the stored data. Recency refers to the time since the last request to the stored data, and frequency refers to the number of requests to the data.

Most of the cache replacement strategies based on recency are variants and extensions of the least recently used strategy (LRU) [30]. The LRU strategy is implemented based on the temporal locality of access to the stored data, and it selects the least recently accessed data to be replaced out of the memory. The principle of the temporal locality can be described as: if the stored data is being accessed, it is likely to be accessed again in the near future. Therefore, the least recently accessed data is a good candidate being replaced. In LRU, the count of data that has not been accessed for the longest time is generally adopted as criteria.

The frequency-based cache replacement strategies are mostly variants and extensions of the least frequently used strategy (LFU) [31]. In the simulation process, the stored data has different frequency values, and these frequency values can be used to instruct the selection of replaced data. LFU strategy sorts these frequency values to find the least frequently used data in the past and performs data replacement. When the power system approaches the stage of equilibrium, the LFU strategy takes advantage of those frequency values to replace the data, because that of a stable state system does not change substantially. However, in general, a large amount of stored data may have the same frequency value, which might lead to unnecessary additional sorting process and increases the extra operating time.

The classic first in first out strategy (FIFO) is also considered in the paper as a reference for comparison.

Since each simulation case has different access characteristics to cache and the hardware specification where the program runs are different, the optimal memory management for EMT simulation is selected with a “fitness for purpose” approach, and it requires a metric to evaluate the performance. LRU, LFU and FIFO strategies are tested in various cases with different cache sizes in Section V.

There are a variety of metrics for evaluating replacement strategies. Commonly used metrics include hit rate, byte hit rate, etc. Supposing R_j is the number of requests for the j^{th} stored data, and in n data is stored in cache, H_j is the number of hits to the j^{th} stored data, S_j is the memory size of the j^{th} storage object, then the hit rate H_r is expressed as:

$$H_r = \sum_{j=1}^{n} H_j \,/ \sum_{j=1}^{n} R_j \tag{9}$$

The hit rate is defined as the percentage of requests that can be satisfied by the cache. And byte hit rate Bh_r is:

$$Bh_r = \sum_{j=1}^{n} S_j H_j \,/ \sum_{j=1}^{n} S_j R_j \tag{10}$$

Hit rate is used in this paper as the metric, because only the number of requests is important to evaluate a cache replacement strategy in EMT simulation, on the other hand, hit rate is a good performance indicator when the cache size is small, most strategies provide similar hit rates when the cache is large [32], and in GPU-based EMT simulation, the stored data is relatively large compared with the cache size.

5. Experimental validations and results

In this section, the wind power plant case in Fig. 7 is tested, where the wind turbine generators are interfaced to the collection grid through back-to-back converters. To verify the GPU-based simulator with caching and memory management, simulation results are compared with that of the CPU-based Matlab program. All programs are tested under Linux operation system, the CPU used in this simulation is Inter(R) Xeon E5-2623 v3, and the GPU is Tesla K20C [29], the details of the GPU are presented in Table 1. The underlying numerical linear algebra operations are carried out with the standard cuBLAS and cuSolverDN libraries. In this paper, all studied cases use the exponential Euler method for numerical integration, although this choice can be made freely according to the priority of different studies. For instance, if the accuracy is taken as the primary consideration, a higher-order method would be adopted.

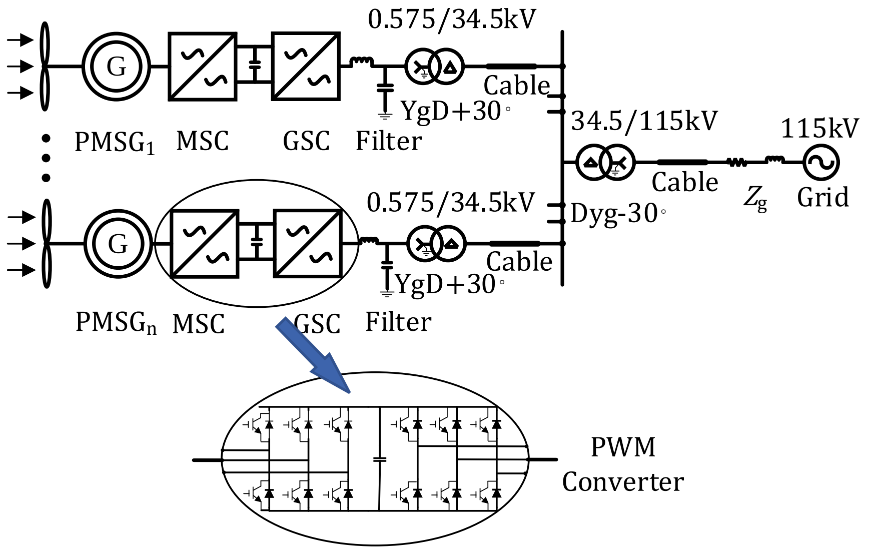

Fig. 7 Wind power plant consisting of N wind turbine generators.

Table 1 Details of NVIDIA K20C

Details	Parameters
Microarchitecture	Kepler
CUDA cores	2496
Streaming multiprocessor	13
GPU clock (MHz)	706
Global memory size (GB)	5
FP64 performance (Tflops)	1.17

Five schemes are compared in this section. Scheme I is the GPU-based matrix exponential method, which calculates φ functions without caching the matrices. It is set as the benchmark to evaluate the performance of other methods. Scheme II tries to caches all φ functions, and

Schemes III, IV and V adopt LRU, LFU, and FIFO algorithms to replace φ functions out of GPU's global memory. The Krylov subspace method is included for comparison purpose only. Several cases of different scales are tested and compared.

The wind power plant consisted of N detailed modeled type-4 wind turbine generators is connected to a collector network [33], each of the converters contains 12 IGBT/anti-parallel diode pairs, modeled as ideal switches with binary resistance, and the whole wind power plant consists of $12N$ switch devices. The time-varying parts of state matrices are update by a partial reformulation process instead of a full matrix update. In the studied cases, up to 10 elements are time-variant for each power converter; they are relatively small in number compared to the size of state matrices. The computation cost of those updating is negligible from practical observations. The parameters of the collector network are presented in the Appendix.

The simulation starts from zero initial state. A fixed 5μs time-step is adopted; the duration of the simulation is 1s. First of all, the correctness of the proposed method is verified. Since the memory management methods do not influence the precision of simulation, all GPU-based schemes have identical waveforms, and the red dotted line of Scheme III results is chosen to compare with the black line of Matlab results. The simulation results of the test wind power plant cases are shown in Fig. 8, including waveforms and comparison between Matlab and CUDA. In Fig. 8 (a-b), the instantaneous current and voltage waveforms of phase A at LCL filter during the time of 0-1.0s show full agreement between the Matlab program and the parallel GPU CUDA program, then the output active power of the wind turbine generator and the voltage of DC-link capacitor between converters are presented in Fig. 8 (c-d) which reach the set values of 1.5MW and 1.5kV. Detailed view of the current waveform at the filter is shown in Fig. 9, it can be observed that the proposed method accurately represents the dynamics of tested cases. The comparison of the simulation results with EMTP in Fig. 10, the WTG filter voltage of phases A is drawn below, and details between 0.174-0.176s are amplified for clearer visualization. It can be observed that the GPU-based EI simulation result is consistent with that of EMTP.

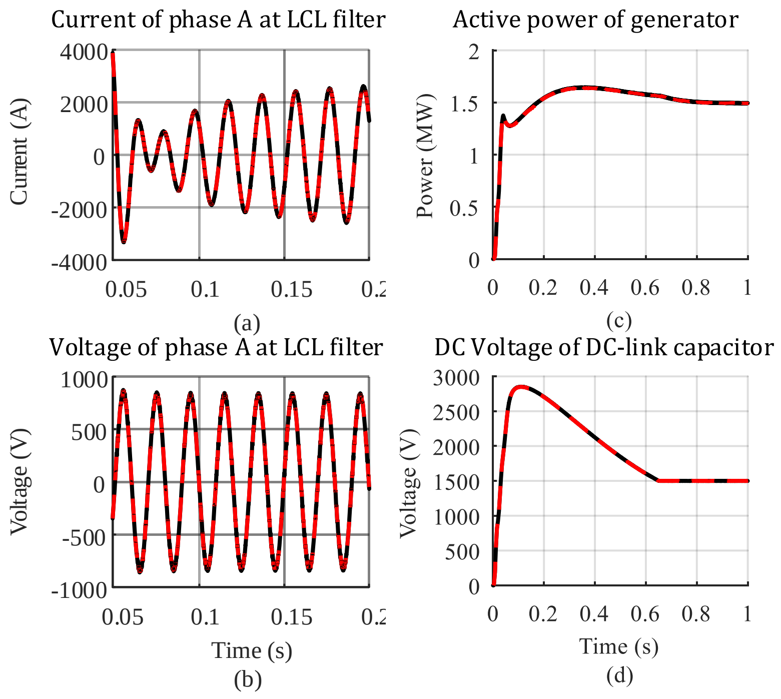

Fig. 8 Simulation results comparison of CPU-(solid black line) and GPU-based (red dashed line) computation.

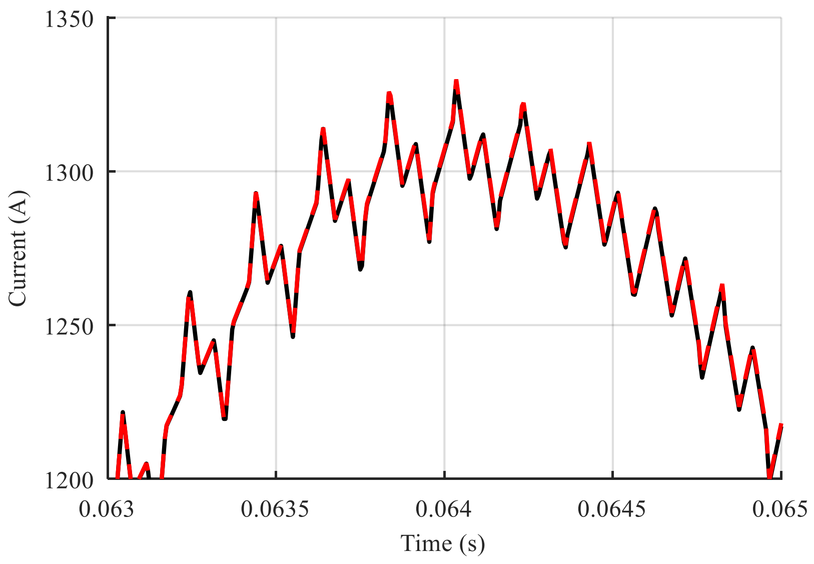

Fig. 9 Detailed view of phase A current waveform at LCL filter, solid black line for CPU result and red dashed line for GPU result.

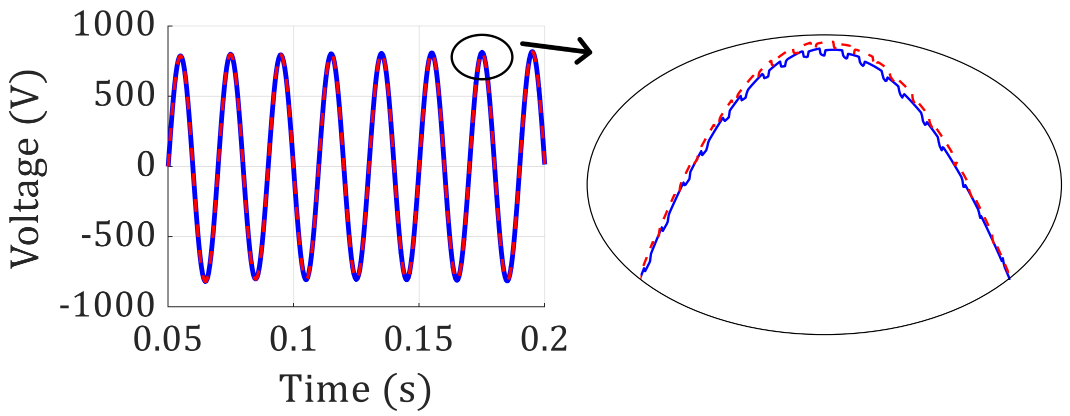

Fig. 10 Simulation result comparison of the proposed algorithm (red dashed line) and EMTP (solid blue line).

With the increasing number of wind turbine generators, the number of on/off state combinations of power electronic switches also increases, causing the reuse of φ functions less frequent. Comparing the computation time summarized in Fig. 11 and Table 2, the speed advantage of Scheme II to V over Scheme I gradually decreased with the number of detailed modeled wind turbine generators increases. It appears that caching is less useful for cases with an enormous number of switches. This is not an issue because the Krylov subspace approach is very efficient in covering these scenarios. It is however interesting to find the explanation behind this phenomenon. The on/off state combination occurrence of the $12N$ power electronic switches is counted during the simulation with the on/off binary string coded in decimal for visual presentation, which leads to Fig. 12 and reveals the pattern of the appearance of switch states, when the number of wind turbine generators changes. The red line in the figure represents the first occurrence of each switch state. The hit rate is represented by the area of purple bar sections minus area under the red line. With an increasing number of generators, the number of power electronic switch states has increased exponentially, and the blue curve has become more flattened. In the meantime results in Table 3 also proves that hit rate of φ functions is declining, as well as the proportion of calculation time saved by caching φ functions along with increasing number of wind turbine generators, which reveals that Scheme II has a certain range of application, beyond which, Scheme II no longer gains acceleration effect. The application of Scheme III, IV and V broadens the application range of Scheme II.

When the GPU global memory size limit is not considered, all φ functions will be stored in cache, none will be replaced out of the memory, then Scheme II will obtain the theoretically ideal speed. When considering the GPU global memory size limit, pre-allocate cache sizes will play an important role in the program, the ratio of the size of φ function to the size of cache is critical to the performance of memory management methods. For this purpose, various cache sizes are pre-allocated to test φ functions hit rates, and then the memory management methods are compared with each other. The following rule is selected to set the pre-allocated cache sizes. Firstly, total φ functions size cached in Scheme I is set as a benchmark, then taking 90%, 80%, all the way to 10% of the benchmark to test the performance of each strategy, and taking 1% of the benchmark to test performance under harsh condition.

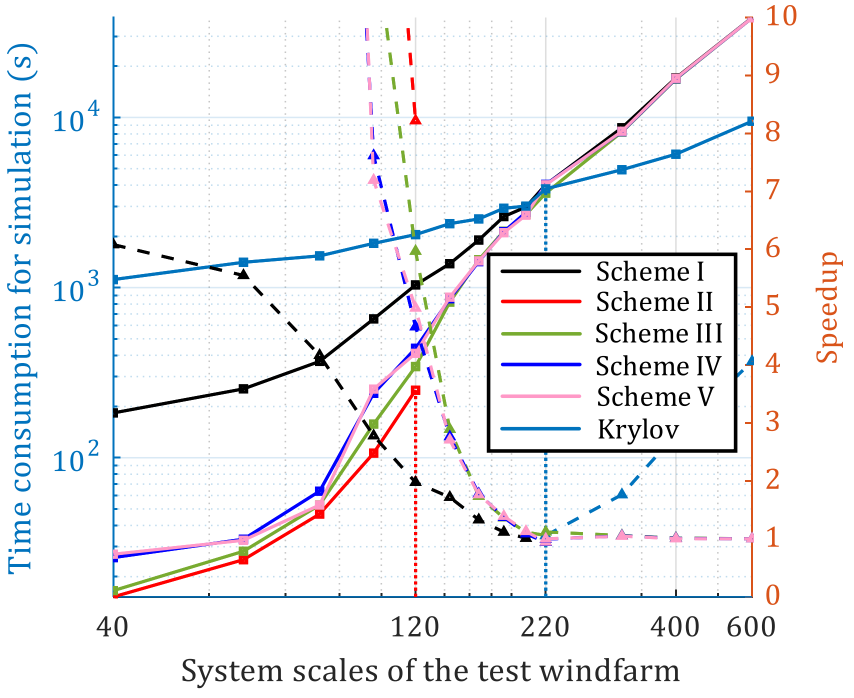

Fig. 11 Simulation time and speedup of the methods, solid line with square represents computation time (left y-axis), dashed line with triangle represents speedup ratio (right y-axis).

Table 2 Simulation time and the speedups compared to the Krylov subspace method

N		24	36	60	120
Scheme I	Time/s	253.89	368.34	1036.85	4127.35
	Speedup	**5.55**	**4.17**	**1.98**	**0.92**
Scheme II	Time/s	25.3	62.1	231.8	/
	Speedup	**55.67**	**24.73**	**8.84**	**/**
Scheme III	Time/s	54.1	302.8	876.1	3531.9
	Speedup	**26.03**	**5.07**	**2.34**	**1.08**
Scheme IV	Time/s	210.1	337.9	938.1	4041.4
	Speedup	**6.70**	**4.55**	**2.18**	**0.94**
Scheme V	Time/s	198.6	338.4	982.3	3939.3
	Speedup	**7.09**	**4.54**	**2.19**	**0.97**
Krylov sub-space	Time/s	1408.4	1535.8	2049.2	3806.8

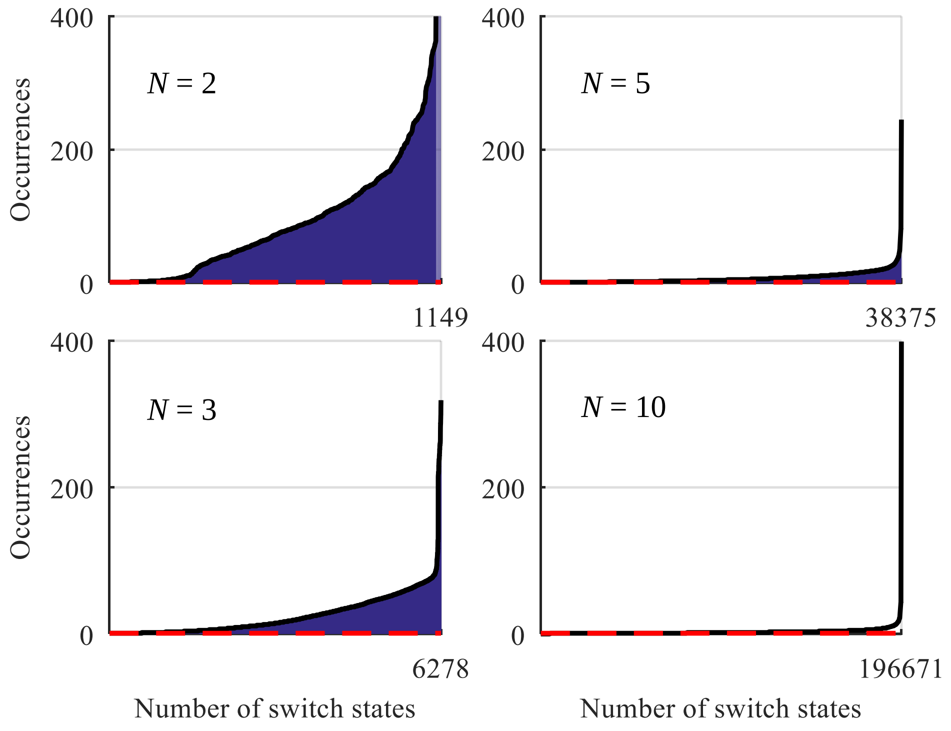

Fig. 12 Statistics of power electronic switch states and occurrence frequency in the tested cases.

Table 3 Hit rate and time saved of Scheme II compared to Scheme I

Cases	Hit rate	Time saved
N=2	98.92%	87.27%
N=3	96.05%	85.28%
N=5	86.01%	76.76%
N=10	64.79%	/

As shown in Fig. 13, LRU and FIFO strategies have almost the same hit rate of $\boldsymbol{\varphi}$ functions when $\boldsymbol{\varphi}$ function size is relatively large compared to the cache size, however when cache sizes become smaller LRU strategy shows better hit rate. Moreover, they outperform LFU strategy on condition that the pre-allocated cache size is relatively small. Although the trend is different, all three curves exhibit saturation, at 80% cache size point, all three curves are saturated, that is, when Schemes III, IV and V are pre-allocated 80% cache size, it is sufficient to achieve fairly similar performance of Scheme I. In addition, as shown with the dotted line, the hit rates of LRU and FIFO strategies reach the level of LFU strategy at 80% size point when 30% cache size is pre-allocated, and it is more than 95%. The phenomenon indicates that the Schemes III, IV and V workflows are able to achieve the desired performance level needing only 30% cache size of Scheme I, which broadened the applicable range of the method proposed in this paper.

In the tested cases, the speedup of each scheme is compared with the slowest scheme. The relationship and comparison of proposed schemes in Fig. 2 are reproduced with case study results in Fig. 11, where N_{d0} =200, N_{d1} =120 and N_{d2} =220-240 (depending on the adopted memory management strategy) are the number of state variables for different cases. Those numbers indicate that when the state variables describing the tested wind power plant case are more than 200, the Krylov subspace method outperforms Scheme I (black), and Scheme II (red) fails at 120-dimensional scenario because of limited GPU global memory. Then these memory management schemes have the best performance in cases that scale from 120 to 220-240 dimensions depending on the adopted memory management strategy. Therefore, the memory management Schemes III-V gain acceleration in between these simulation scales and become more efficient for EMT simulation of power electronic converters. Obviously, Scheme I is more suitable in smaller cases. Schemes III, IV and V adopted cache sizes listed in Table 4 for test, they gain similar but less speedup than Scheme II, in addition, when Scheme II failed, those schemes still gain acceleration, they are faster than Scheme I, so memory management methods broaden the range of application of GPU-based matrix exponential method, as listed in Table 2, Scheme III gains 1.2x speedup when wind power plant consisting of 10 wind turbine generators. In the meantime, LRU strategy shows more remarkable performance in nearly all conditions, it outperforms LFU and FIFO strategy when the number of wind turbine generators is less than 6 and more than 10, even in between the range, the performances among these three strategies are quite similar. Therefore, the LRU strategy is chosen for EMT simulation memory management.

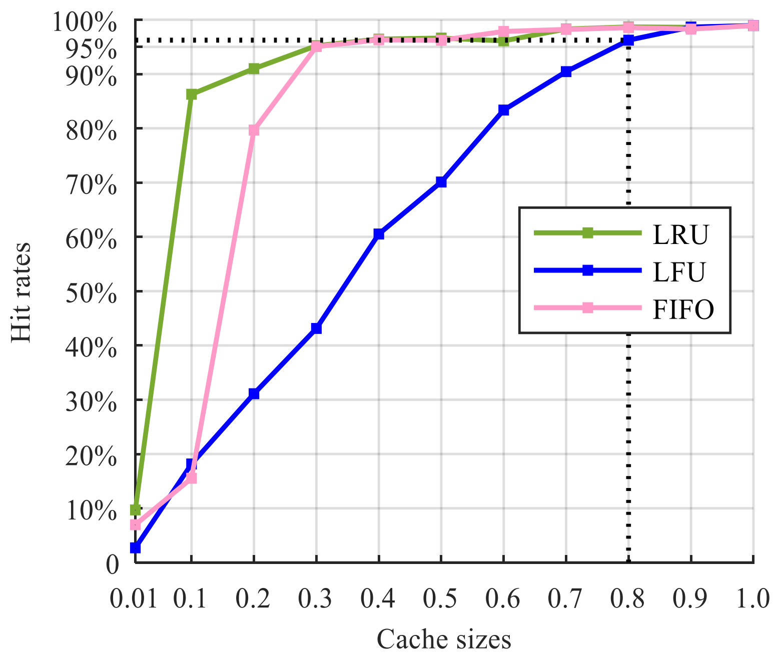

Fig. 13 Performance of FIFO, LRU and LFU strategies in different cache sizes.

Table 4 Size of $\boldsymbol{\varphi}$ function and pre-allocated cache sizes in the tested case

Cases	Size of $\boldsymbol{\varphi}$ function (MB)	Size of cache (MB)
N=2	0.027	25.22
N=3	0.049	244.61
N=5	0.110	3373.59
N=10	0.369	4096.00

6. Conclusion

This paper designs a GPU-based parallel exponential integration algorithm for efficient EMT simulations. By offloading computationally intensive matrix exponential functions to GPU, the simulation is accelerated. Next, we cache $\boldsymbol{\varphi}$ function on GPU memory to cope with the high-frequency time-varying state matrices in power converter EMT simulation, it shows good performance in a certain range and the gained performance fades when the number of switches keeps growing large. At last, the proposed memory management method organizes the limited cache space from different perspectives, three cache replacement strategies are tested and compared and the hit rates indicate that the LRU strategy is the most suitable memory management for power converter transient simulation. A detailed modeled wind power plant is tested to verify the efficiency and accuracy of the proposed method. Results show that the proposed method program achieves considerable speedups over the traditional direct method program, and on account of its memory management, the program retains speedups for larger-scale EMT simulation with many power converters.

The test cases in this paper are of medium-scale, the proposed methods gain good performance. Krylov subspace is more suitable to be applied to large-scale cases, and there are ways to improve the efficiency of power system EMT simulation consisting of power converters like multirate method. Those algorithms integrated with the caching method can be explored. In future research, these directions will be investigated along with the GPU utilization for the simulation of large-scale power converter.

Acknowledgements

This work was supported in part by the National Natural Science Foundation of China under Grant 51807131 and U1866207, and in part by the China Postdoctoral Science Foundation under Grant 2018M641646.

References

[1] Wang, X., Freitas, W., Xu, W., et al.: 'Impact of DG interface controls on the Sandia frequency shift ant islanding method', *IEEE Trans. Energy Convers.*, 2007, **22**, (3), pp. 792–794

[2] An, Z., Shen, C., Zheng, Z.T., et al.: 'Scenario-based analysis and probability assessment of sub-synchronous oscillation caused by wind farms with direct-driven wind generators', *J. Mod. Power Syst. Clean Energy*, 2019, **7**, (02), pp. 243-253

[3] Dommel, H.W.: 'Electromagnetic Transients Program Reference Manual' (Bonneville Power Administration, Portland, OR, 1986)

[4] Peralta, J., Saad, H., Dennetière, S., et al.: 'Detailed and averaged models for a 401-level MMC-HVDC system', *IEEE Trans. Power Deliv.*, 2012, **27**, (3), pp. 1501–1508

[5] Falcao, D.M., Kaszkurewicz, E., Almeida, H.S.: 'Application of parallel processing techniques to the simulation of power system electromagnetic transients', *IEEE Trans. Power Syst.*, 1993, **8**, (1), pp. 90-96

[6] Armstrong, M., Marti, J.R., Linares, J.L., et al.: 'Multilevel MATE for Efficient Simultaneous Solution of Control Systems and Nonlinearities in the OVNI Simulator'. *Proc. IEEE PES General Meet.*, 2007, pp. 1-1

[7] Strunz, K., Carlson, E.: 'Nested fast and simultaneous solution for time domain simulation of integrative power electric and electronic systems', *IEEE Trans. Power Deliv.*, 2007, **22**, (1),

pp. 277-287

[8] Mu, Q., Liang, J., Zhou, X., et al.: ‘A node splitting interface algorithm for multi-rate parallel simulation of DC grids’, *CSEE J. Power Energy Syst.*, 2018, **4**, (3), pp. 388-397

[9] Chiniforoosh, S., Jatskevich, J., Yazdani, A., et al.: ‘Definitions and applications of dynamic average models for analysis of power systems’, *IEEE Trans. Power Deliv.*, 2010, **25**, (4), pp. 2655-2669

[10] Henschel, S. ‘Analysis of electromagnetic and electromechanical power system transients with dynamic phasors’. PhD dissertation, University of British Columbia, 1999

[11] Abusalah, A., Saad, O., Mahseredjian, J., et al.: ‘CPU based parallel computation of electromagnetic transients for large power grids’, *Electr. Power Syst. Res.*, 2018, **160**, pp. 57-63

[12] Liu, C., Ma, R., Bai, H., et al.: ‘A new approach for FPGA-based real-time simulation of power electronic system with no simulation latency in subsystem partitioning’, *Int. J. Electr. Power Energy Syst.,* 2014, **63**, pp. 285-292

[13] Kuffel, R., Giesbrecht, J., Maguire, T., et al.: ‘RTDS-a fully digital power system simulator operating in real time’. *Proc. First Int. Conf. Digital Power System Simulators*, April 1995, pp. 19

[14] Bélanger, J., Snider, L.A., Paquin, J.N., et al.: ‘A modern and open real-time digital simulator of contemporary power systems’. *Proc. Int. Conf. on Power Systems Transients*, June 2009, pp. 1–10

[15] Zhang, C., Xu, Y., Chen, Y., et al.: ‘Batch Computing Method for Sensitivity Analysis of Large Power Grids Based on GPU Acceleration’. *IEEE International Electrical and Energy Conference*, 2019

[16] Debnath, J.K., Fung, W.K. Gole, A.M., et al.: ‘Simulation of large-scale electrical power networks on graphics processing units’, *Proc. IEEE EPEC*, 2011, pp. 284–289

[17] Lin, N., Dinavahi, V.: 'Variable Time-Stepping Modular Multi-Level Converter Model for Fast and Parallel Transient Simulation of Multi-Terminal DC Grid', *IEEE Trans. Ind. Electron.*, 2018, **66**, (9), pp. 6661-6670

[18] Zhou, Z., Dinavahi, V.: 'Fine-Grained Network Decomposition for Massively Parallel Electromagnetic Transient Simulation of Large Power Systems', *IEEE Power Energy Technol. Syst. J.*, 2017, **4**, (3), pp. 51-64

[19] Y., Chen, Y., Yu, Z., et al.: 'CloudPSS: A High-Performance Power System Simulator Based on Cloud Computing', arXiv preprint arXiv:1903.01081

[20] Song, Y., Chen, Y., Huang, S., et al.: 'Efficient GPU-based electromagnetic transient simulation for power systems with thread-oriented transformation and automatic code generation', *IEEE Access*, 2018, **6**, pp. 25724-25736

[21] Song, Y., Chen, Y., Huang, S., et al. 'Fully GPU-based electromagnetic transient simulation considering large-scale control systems for system-level studies', *IET Gener., Transmiss., Distrib.*, 2017, **11**, (11), pp. 2840-2851

[22] Wang, C., Fu, X., Li, P., et al.: 'Accurate dense output formula for exponential integrators using the scaling and squaring method', *Appl. Math. Lett.*, 2015, **43**, pp. 101-107

[23] Wang, C., Fu, X., Li, P., et al.: 'Multiscale simulation of power system transients based on the matrix exponential function', *IEEE Trans. Power Syst.*, 2017, **32**, (3), pp. 1913-1926

[24] Wang, C., Fu, X., Li, P., et al.: 'Matrix exponential based electromagnetic transients simulation algorithm with Krylov subspace approximation and accurate dense output', *Proc. IEEE PES General Meet.*, 2014, pp. 1–5

[25] Fu, X., Wang, C., Li, P., et al.: 'Exponential integration algorithm for large-scale wind farm simulation with Krylov subspace acceleration', *Appl. Energy*, 2019, **254**

[26] Fu, X., Seye, S. M., Mahseredjian. J., et al.: 'A comparison of numerical integration methods

and discontinuity treatment for EMT simulations', *Power Systems Computation Conference*, 2018, pp: 1-7

[27] Saad, Y.: 'Analysis of some Krylov subspace approximations to the matrix exponential operator', *SIAM J. Numer. Anal.*, 1992, **29**, (1), pp. 209-228

[28] NVIDIA: 'CUDA c programming guide 8.0' (NVIDIA Corporation, Santa Clara, CA, USA, 2017)

[29] Lindholm, E., Nickolls, J., Oberman, S., et al.: 'NVIDIA Tesla: A Unified Graphics and Computing Architecture', *IEEE Micro*, 2008, **28**, (2), pp. 39-55

[30] Che, L., Qiu, B.: 'Landmark LRU: An efficient scheme for the detection of elephant flows at internet routers', *IEEE Commun. Lett.,* 2006, **10**, (7), pp. 567–569

[31] Sokolinsky, L.: 'LFU-K: an effective buffer management replacement algorithm', *Lect. Notes Comput. Sci.*, 2004, **2973**, pp. 670–681

[32] Wong, K.: 'Web cache replacement policies: A pragmatic approach', *IEEE Netw.*, 2006, **20**, (1), pp. 28–34

[33] Hussein, D.N., Mahmoud, M., Iravani, R.: 'A type-4 wind power plant equivalent model for the analysis of electromagnetic transients in power systems', *IEEE Trans. Power Syst.*, 2013, **28**, (3), pp. 3096–3104

Appendices

The 115 kV grid impedance: 0.1448 + j2.897 Ω.

The wind power generator:

a) moment of inertia = 100000 kg·m^2;

b) number of pole pairs = 120;

c) diameter of turbine blades = 34 m;

d) wind speed = 12 m/s;

e) stator resistant = 0.0081 Ω;

f) permanent flux = 2.458 Wb.

The 115 kV cable PI model:

a) resistance = 0.3815 Ω/km;

b) inductance = 0.4 mH/km;

c) conductance = 0.08 μF/km;

d) length = 2km.

The step-up transformer:

a) nominal power = 10.0 MVA;

b) winding impedance = 0.00083 + j0.025 p.u.

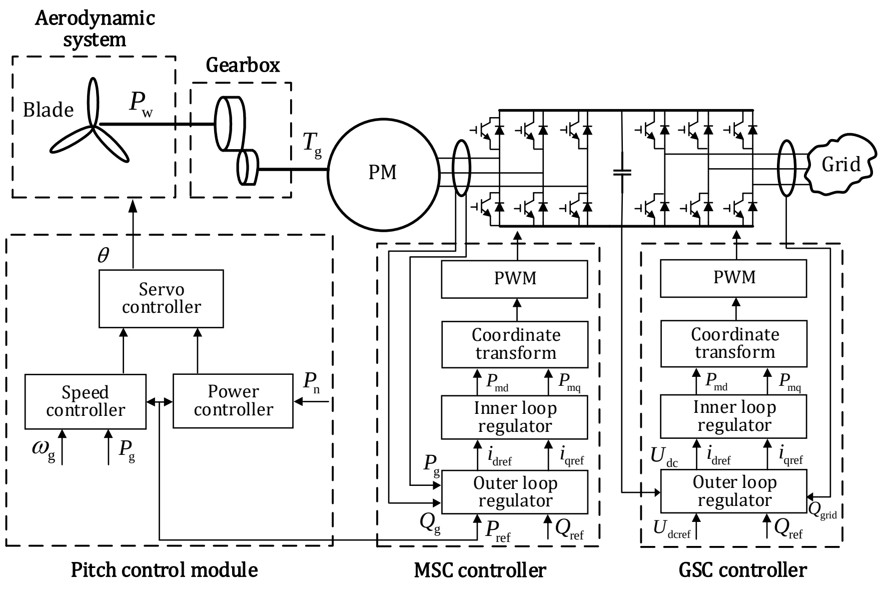

Fig. 14 The grid-connected control system of wind farm.

In terms of converter control strategy, the MSC controller adopts dual-loop vector control to realize the decoupling control of the active power and reactive power of the generator, and the GSC controller adopts grid converter voltage reference frame control.