Parallel Computation of Wrench Model for Commutated Magnetically Levitated Planar Actuator

Fengqiu Xu, *Student Member, IEEE*, Venkata Dinavahi, *Senior Member, IEEE*, and Xianze Xu

***Abstract*—An accurate wrench model is significant for the simulation, manufacture, and control of the commutated magnetically levitated planar actuator (CMLPA). With plenty of coils and permanent magnets employed in the coil set and magnet array of CMLPA, the computational burden of the corresponding wrench model can be substantial. This paper proposes an accurate, universal, and robust parallel massive-thread wrench model (PMWM) for the CMLPA. In PMWM, the magnetic node, Gaussian quadrature, and coordinate transformation are employed to express the interaction between magnet array and coil set. All of these calculation modules are implemented on the graphics processing unit in a massively parallel framework by CUDA. In order to highlight the performance of this PMWM, the wrench model of three different CMLPAs is computed. The computation accuracy and efficiency of proposed PMWM are compared with the finite-element method software Ansys Maxwell and a boundary element method software package named Radia, respectively. The same wrench model is also implemented on a multicore CPU through OpenMP and the comparative results are presented to show significant acceleration of the proposed massive-thread model.**

***Index Terms*—Commutated magnetically levitated planar actuator (CMLPA), graphics processing unit (GPU), parallel processing, wrench model.**

Nomenclature

N, M	Coils and magnets amount.
T	Magnetic nodes amount.
i, q	Coil and coil segment indices.
j, k	Magnet and magnetic node indices.
g_1, g_2, g_3	Gaussian quadrature sequence numbers.
τ_c, τ_m	Coil pitch and magnet pitch.
H_m	Cuboid or cylindrical magnet height.
L_m, W_m	Cuboid magnet length and width.
R_m	Cylindrical magnet radius.
H_c, L_c, W_c	Coil length, width, and height.
R_{cin}, R_{cout}	Coil inner radius and outer radius.

Manuscript received January 6, 2016; revised April 21, 2016; accepted June 14, 2016. Date of publication July 19, 2016; date of current version November 8, 2016. This work was supported in part by the Natural Science and Engineering Research Council of Canada and in part by the Open Research Fund of the Key Laboratory of Spectral Imaging Technology of the Chinese Academy of Sciences.

F. Xu and V. Dinavahi are with the Department of Electrical and Computer Engineering, University of Alberta, Edmonton, AB T6G 2V4, Canada (e-mail: fengqiu@ualberta.ca; dinavahi@ualberta.ca).

X. Xu is with the Electrical Information School, Wuhan University, Wuhan 430072, China (e-mail: xuxianze@whu.edu.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIE.2016.2592866

I. Introduction

COMMUTATED magnetically levitated planar actuator (CMLPA), employing large-scale coil set and magnet array in the stator and translator, respectively, provides multiple degrees of freedoms (DOFs), long stroke, and high-resolution positioning [1]–[4]. Since the stator and translator in the CMLPA are contactless, there is no friction, stiction, and backlash, resulting in a simple dynamic system, excellent repeatability, high positioning accuracy, large control bandwidth, and clean working environment. All these features make CMLPA a very attractive in modern industrial application, where high-precision motion control is necessary [5]–[7]. To design, control, and emulate the CMLPA, an accurate computation of the wrench matrix containing the force and torque acting on the magnet array due to each coil is vital [8]–[11]. In the literature, following are various analytical methods to solve the wrench model in the magnetic actuator:

1) harmonic method [12]–[15];
2) magnetic charge method [15]–[19];
3) magnetic vector potential method [20], [21]; and
4) equivalent magnetic circuit method [22]–[25].

In the harmonic method, the current carrying region and permanent magnets should be spaced periodical to reduce the effect of high harmonics, and the area of the airgap should be large enough to decrease the impact of boundary effects. In the magnetic charge method or magnetic vector potential method, the force and torque are expressed as multiple integrals; as such this method cannot always result in the analytical expression and it requires a compromise between the computational load and the calculation accuracy. In the equivalent magnetic circuit method, the force and torque are obtained by computing the derivative of magnetic coenergy related to the translation and rotation; it is not suitable here because the CMLPA used in high-precision motion control always employs ironless coils to prevent nonlinearity, eddy-current loss, and hysteresis. In general, these methods are effective for some CMLPAs with specific design structures when the translator and stator are aligned. They become time consuming or even unavailable under realistic conditions where the pitch, yaw, and roll of translator exist. Under these circumstances, the finite-element method (FEM) [26], [27] or the boundary-element method (BEM) [28], [29] which provide a robust solution for the force and torque distribution are viable alternatives, but the execution time can be

prohibitive if we want to obtain an accurate solution. However, thanks to the developments in the current parallel compute technologies, the execution time can be significantly decreased with high computation accuracy on parallel architecture compute engine.

Currently, plenty of work about the usage of parallel computation for the solution of the general magnetostatic problem is focused on the finite-difference method (FDM), FEM, and BEM [30]–[32]. The multiple instruction, multiple data method is employed to achieve parallelism because these numerical techniques consist of several computation phases for the iterations until a convergent solution is obtained. The programing of each phase should be designed carefully to eliminate the unnecessary overhead. However, based on the magnetic charge or vector potential method, much research only takes advantage of the numerical integral to solve the magnetostatic problem, while little work was done to explore the inherent parallelism. Furthermore, the superposition law, a basic principle of electromagnetic problems, can help to obtain a fine-grained parallelism and reduce the phases of computation. In this case, single instruction, multiple data (SIMD) can be employed, which can result in very low overhead.

In this study, we build the wrench model for the CMLPA via three tools: magnetic node [33], Gaussian quadrature [34], and coordinate transformation. The magnetic node describes the field around a permanent magnet based on the magnetic charge method. The Gaussian quadrature is employed to simplify the Lorenz integral for force and torque computation. Through coordinate transformation, the force and torque are solved regardless of the relative orientation between the magnet array and coil set. We exploit three levels of parallelism.

1) Each coil in the coil set and each magnet in the magnet array are independent.
2) The field created by a magnet is expressed as the sum of the effects from several independent sources.
3) The Lorenz integral equation is solved by summing several independent expressions via Gaussian quadrature.

The parallel massive-thread wrench model (PMWM) is mapped to the massively parallel architecture of the graphics processing unit (GPU), which has native parallel many-core processing units and high-performance floating-point number processors [35]. The GPU has been exploited for accelerating many applications, such as immersive stereoscopic display [36], visual servo control [37], transient stability and electromagnetic transient simulation [38]–[40], and parameters estimation of electrical machine [41], and power systems [42], [43]. The general-purpose parallel computing environment, CUDA, introduced by NVIDIA was used to develop these applications.

The rest of this paper is organized as follows. Section II introduces the construction of PMWM utilizing the aforementioned methods. Section III demonstrates the parallel massive-thread component implementation on the GPU. In Section IV, utilizing three case studies of CMLPA, we compare the results achieved by the proposed method against Ansys Maxwell, Radia, and with the same model implemented on multicore CPU through OpenMP to show the computation accuracy and efficiency. Section V gives the conclusions of this paper.

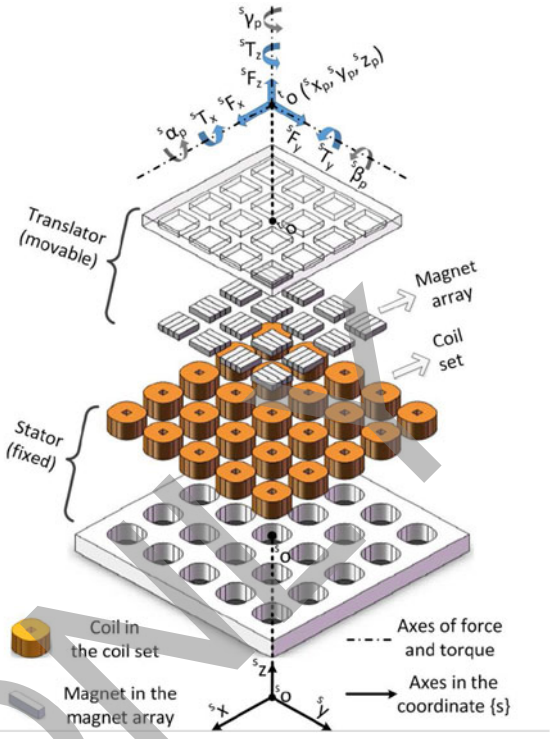

Fig. 1. Exploded view of a typical CMPLA with the force and torque acting on the translator.

II. Massively Parallel Wrench Model Computation

A. Wrench Model

A typical CMLPA that contains the stator and the translator is shown in Fig. 1. A wrench vector, ${}^s\mathbf{w}$, including three forces and three torques acting on the translator, and a position and orientation vector (POV), ${}^s\mathbf{p}$, including the translation and rotation of the translator, are defined in the stator coordinate system $\{s\}$

$$ {}^s\mathbf{w} = ({}^sF_x\ {}^sF_y\ {}^sF_z\ {}^sT_x\ {}^sT_y\ {}^sT_z)^{\mathrm{T}}, \tag{1} $$

$$ {}^s\mathbf{p} = ({}^sx_p\ {}^sy_p\ {}^sz_p\ {}^s\alpha_p\ {}^s\beta_p\ {}^s\gamma_p)^{\mathrm{T}}. \tag{2} $$

In Fig. 1, point ${}^so$, the origin of stator coordinate $\{s\}$, is coincided with the geometric center of the top surface of stator, and point ${}^to\,({}^sx_p, {}^sy_p, {}^sz_p)$ is coincided with the mass center of the translator. The rotation variables, ${}^s\alpha_p, {}^s\beta_p, {}^s\gamma_p$, in ${}^s\mathbf{p}$ and torque elements, ${}^sT_x, {}^sT_y, {}^sT_z$, in ${}^s\mathbf{w}$ are due to ${}^to$. The wrench vector ${}^s\mathbf{w}$ and current vector $\mathbf{I}$ meet the relationship given in (3), and wrench matrix is written as (4)

$$ {}^s\mathbf{w} = \mathbf{\Gamma}({}^s\mathbf{p}) \cdot \mathbf{I} = \mathbf{\Gamma}({}^s\mathbf{p}) \cdot (I_1 \quad \cdots \quad I_i \quad \cdots \quad I_N)^{\mathrm{T}}, \tag{3} $$

$$ \mathbf{\Gamma}({}^s\mathbf{p}) = \begin{pmatrix} {}^s\mathrm{F}_{1,x}({}^s\mathbf{p}) & \cdots & {}^s\mathrm{F}_{i,x}({}^s\mathbf{p}) & \cdots & {}^s\mathrm{F}_{N,x}({}^s\mathbf{p}) \\ {}^s\mathrm{F}_{1,y}({}^s\mathbf{p}) & \cdots & {}^s\mathrm{F}_{i,y}({}^s\mathbf{p}) & \cdots & {}^s\mathrm{F}_{N,y}({}^s\mathbf{p}) \\ {}^s\mathrm{F}_{1,z}({}^s\mathbf{p}) & \cdots & {}^s\mathrm{F}_{i,z}({}^s\mathbf{p}) & \cdots & {}^s\mathrm{F}_{N,z}({}^s\mathbf{p}) \\ {}^s\mathrm{T}_{1,x}({}^s\mathbf{p}) & \cdots & {}^s\mathrm{T}_{i,x}({}^s\mathbf{p}) & \cdots & {}^s\mathrm{T}_{N,x}({}^s\mathbf{p}) \\ {}^s\mathrm{T}_{1,y}({}^s\mathbf{p}) & \cdots & {}^s\mathrm{T}_{i,y}({}^s\mathbf{p}) & \cdots & {}^s\mathrm{T}_{N,y}({}^s\mathbf{p}) \\ {}^s\mathrm{T}_{1,z}({}^s\mathbf{p}) & \cdots & {}^s\mathrm{T}_{i,z}({}^s\mathbf{p}) & \cdots & {}^s\mathrm{T}_{N,z}({}^s\mathbf{p}) \end{pmatrix}. \tag{4} $$

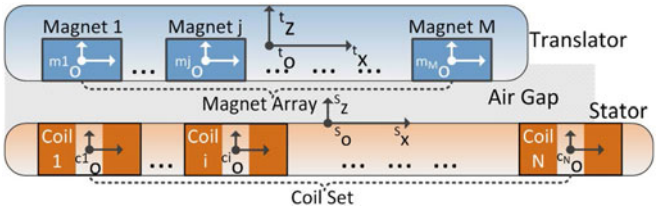

Fig. 2. Lumped topology structure of CMLPA.

$[{}^s\mathbf{F}_i, {}^s\mathbf{T}_i]^{\mathrm{T}}$, the ith column of the wrench matrix, is the wrench vector due to coil i. ${}^s\mathbf{F}_i$ and ${}^s\mathbf{T}_i$ can be obtained by superposition theorem as follows:

$$[{}^s\mathbf{F}_i({}^s\mathbf{p}), {}^s\mathbf{T}_i({}^s\mathbf{p})]^{\mathrm{T}} = \sum_{j=1}^{M} [{}^s\mathbf{F}_{ij}({}^s\mathbf{p}), {}^s\mathbf{T}_{ij}({}^s\mathbf{p})]^{\mathrm{T}} \quad (5)$$

where ${}^s\mathbf{F}_{ij}({}^s\mathbf{p})$ and ${}^s\mathbf{T}_{ij}({}^s\mathbf{p})$ represents the force and torque acting on the magnet j due to coil i.

B. Interaction Between Single Coil and Single Magnet

In order to analyze this issue, translator coordinate system $\{t\}$, magnet coordinate system $\{m_j\}$, and coil coordinate system $\{c_i\}$ are defined in the lumped topology structure of CMLPA as shown in Fig. 2. The origins, ${}^t o$, ${}^{m_j} o$, and ${}^{c_i} o$, are the mass center of the translator, magnet j, and coil i, respectively. The directions of coordinate axes in $\{m_j\}$, $\{c_i\}$ are dependent on the orientation of the corresponding magnet and coil, while the directions of coordinate axes in $\{t\}$ are decided by the rotation variables in POV ${}^s\mathbf{p}$.

The force and torque acting on magnet are able to be solved in coil coordinate system $\{c_i\}$ due to the Lorentz integrals, which are the reaction acting on the current carrying coil

$${}^{c_i}\mathbf{F}_{ij}({}^s\mathbf{p}) = -\iiint_{\mathrm{Vcoil}_i} {}^{c_i}\mathbf{J} \times {}^{c_i}\mathbf{B}({}^s\mathbf{p}) dV, \quad (6)$$

$${}^{c_i}\mathbf{T}_{ij}({}^s\mathbf{p}) = -\iiint_{\mathrm{Vcoil}_i} {}^{c_i}\mathbf{r}({}^s\mathbf{p}) \times {}^{c_i}\mathbf{J} \times {}^{c_i}\mathbf{B}({}^s\mathbf{p}) dV \quad (7)$$

where ${}^{c_i}\mathbf{r}({}^s\mathbf{p})$ is the moment arm from the volume element to the point ${}^t o$; ${}^{c_i}\mathbf{B}({}^s\mathbf{p})$ is the magnetic flux density created by magnet j; ${}^{c_i}\mathbf{J}$ is the current density in the coil i. In general, the relative position and orientation between the coil and magnet are arbitrary, and these volume integrals are hard to get analytical expression. Magnetic node, Gaussian quadrature, and coordinate transformation can help to simplify (6) and (7).

1) Magnetic Node: This method transforms the magnetic field around permanent magnet to the effect of several independent sources, named magnetic nodes, which are presented by points ${}^{m_j} D_k$ in Fig. 3(a). The magnetic flux density at point ${}^{m_j} G$ resulting from a cuboid or cylindrical permanent magnet is the sum of effect from each magnetic node:

$${}^{m_j}\mathbf{B}({}^{m_j}\mathbf{G}) = \sum_{k=0}^{T} {}^{m_j}\mathbf{B}_k({}^{m_j}\mathbf{G} - {}^{m_j}\mathbf{D}_k). \quad (8)$$

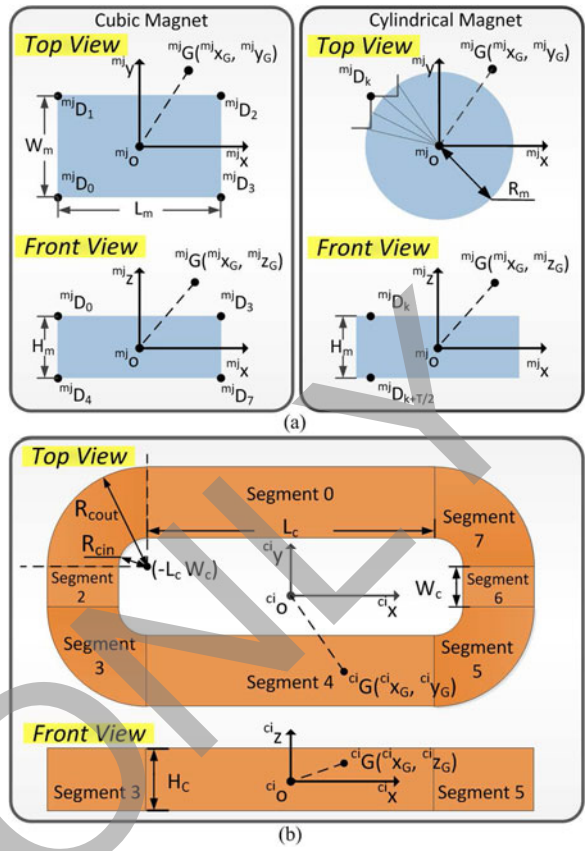

Fig. 3. (a) Magnetic nodes in cuboid and cylindrical magnets. (b) Segments in a typical coil.

T is equal to 8 in the cuboid magnet, and we assume T is 128 in the cylindrical one [33].

2) Gaussian Quadrature: An accurate result of integral can be obtained if the integrated function is continuous in the integration range. Based on the error estimate method proposed in [34], we utilize the eight-order rule here as given by

$$\int_{x_1}^{x_2} f(x)\, dx = \frac{x_2 - x_1}{2} \sum_{g=0}^{7} w_g f\left(\frac{x_2 + x_1}{2} + \lambda_g \frac{x_2 - x_1}{2}\right) \quad (9)$$

where λ_g and w_g is the node and weight of the Gaussian quadrature. Therefore, (6) and (7) are expressed as the triple summations. Since a typical coil is divided into eight segments shown in Fig. 3(b), the region of integral is divided into eight parts. Such division enables definite integrals, so that the Gaussian nodes are mapped to the real Gauss-nodes ${}^{c_i} G$ located in the coil. Under this circumstance, it is nonblocking to solve the Gaussian quadrature because the integration limits of the inner and outer integral are independent.

3) Coordinate Transformation: The vectors, including the magnetic flux density, the current density, and the wrench matrix, can be transformed from one coordinate to another by a transforming matrix $\mathbf{R}$ and a translating vector $\mathbf{t}$. Thus, the

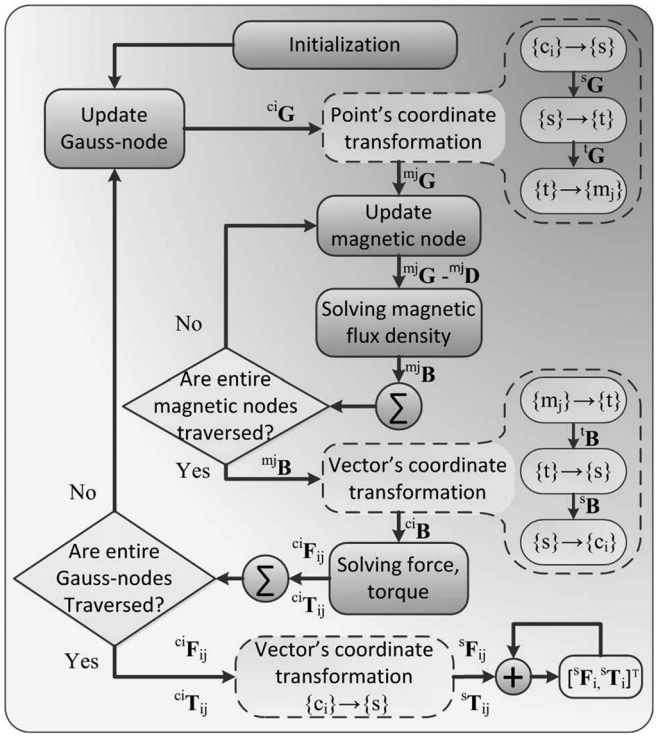

Fig. 4. Flowchart for force and torque calculation between single coil and magnet.

interaction between single coil and magnet are

$$ {}^{s}\mathbf{F}_{ij} = -\sum_{q=0}^{7}\sum_{g_1=0}^{7}\sum_{g_2=0}^{7}\sum_{g_3=0}^{7}\sum_{k=0}^{2T} w_{g_1} w_{g_2} w_{g_3} $$

$$ {}^{s}_{c_i}\mathbf{R} \cdot {}^{c_i}\mathbf{J} \times {}^{c_i}_{m_j}\mathbf{R} \cdot {}^{m_j}\mathbf{B}_k \left({}^{m_j}_{c_i}\mathbf{R} \cdot \left({}^{c_i}\mathbf{G} - {}^{m_j}_{c_i}\mathbf{t} \right) - {}^{m_j}\mathbf{D} \right), \quad (10) $$

$$ {}^{s}\mathbf{T}_{ij} = \sum_{q=0}^{7}\sum_{g_1=0}^{7}\sum_{g_2=0}^{7}\sum_{g_3=0}^{7}\sum_{k=0}^{2T} w_{g_1} w_{g_2} w_{g_3} {}^{s}_{c_i}\mathbf{R} \cdot {}^{t}_{c_i}\mathbf{R} \cdot $$

$$ \left({}^{c_i}\mathbf{G} - {}^{t}_{c_i}\mathbf{t} \right) \cdot {}^{c_i}\mathbf{J} \times {}^{c_i}_{m_j}\mathbf{R} \cdot {}^{m_j}\mathbf{B}_k \left({}^{m_j}_{c_i}\mathbf{R} \cdot \left({}^{c_i}\mathbf{G} - {}^{m_j}_{c_i}\mathbf{t} \right) - {}^{m_j}\mathbf{D} \right). \quad (11) $$

We obtain ${}^{s}_{c_i}\mathbf{R}$, ${}^{s}_{c_i}\mathbf{t}$, ${}^{m_j}_{t}\mathbf{R}$, and ${}^{m_j}_{t}\mathbf{t}$ due to the design parameters of the magnet array and coil set, while calculate ${}^{s}_{t}\mathbf{R}$ and ${}^{s}_{t}\mathbf{t}$ due to POV ${}^{s}\mathbf{p}$.

III. Implementation of Massive-Thread Wrench Model

A. Typical Implementation in Single-Thread Processor

With magnetic node, Gaussian quadrature, and coordinate transformation, the interaction between the coil set and the magnet array under arbitrary relative position and orientation is solved through the flowchart shown in Fig. 4. At the end of the flowchart, the result is added to the corresponding memory where the specific element of wrench matrix is stored. In the normal single-thread program, $N \times M$ times main loop is necessary for the implementation, while two *for* loops are also required for traversing the magnetic nodes and Gauss-nodes, respectively, in the main loop.

B. Extraction of Parallelism

Based on the aforementioned algorithm, the PMWM can employ the fine-grained type of parallelism as shown in Fig. 5. Each branch represents the procedure to calculate the smallest cell of force and torque. One branch in this figure contains two starting points and one ending point: one starting point locates on the coil set; the other starting point locates on the magnet array; the ending point locates on the $\sum$ symbol. Branch 0 is a typical branch shown in dotted line. As branches in Fig. 5 are independent with others, the PMWM is data parallel, and suitable to be realized on the SIMD-based architectures such as the GPU for computation. Each branch represents one iteration in the loop in Fig. 4, and is executed in an individual thread without divergence.

Fig. 6 describes the flow of each thread implementing one branch. There are two phases in the computation processing as depicted. In the first phase, being similar with Fig. 4, the threads need to obtain the Gauss-nodes and magnetic nodes, solve the magnetic flux density, calculate the force and torque, and execute the coordinate transformation, but do not need to implement decision or jump instruction in the processing. In the second phase, the reduction and atomic add operation are used to realize the summation depicted as the last step of processing in Fig. 5. The result of the atomic operation will be stored in different memories based on the index of coil.

In Fig. 5, there are $8^4 \times T \times N \times M$ branches, namely different threads, in the PMWM. The parameters for the calculation in each thread are collected due to these variables, i, j, q, g_1, g_2, g_3, k. The description, range, and weight of these variables are described in Table I. We define an index-array $\mathbf{v} = \{i, j, q, g_1, g_2, g_3, k\}$, and its value, val_v, represents the index of each thread is calculated as

$$ val\_v = T \cdot 8^4 \cdot N \cdot i + T \cdot 8^4 \cdot j + T \cdot 8^3 \cdot q + T \cdot 8^2 \cdot g_3 + T \cdot 8^1 \cdot g_2 + T \cdot g_3 + k. \quad (12) $$

C. Implementation in Massive-Thread Architecture

The Kepler architecture of GPU from NVIDIA is chosen for implementing the parallel massive-thread program, and CUDA is the programing tool used to implement the PMWM. A CUDA program separates the hardware resources into two parts: CPU side as the host, on which the serial main function runs; the GPU side as the device, on which the parallel kernel function runs. The flow chart of main function and kernel function are shown in Fig. 7.

1) Main Function on CPU Side: The main function which controls the data stream runs in a serial mode and contains four steps as described as follows:

1) for the initialization, writing the rotating matrix and translating vector, ${}^{s}_{c_i}\mathbf{R}$, ${}^{m_j}_{t}\mathbf{R}$, ${}^{s}_{c_i}\mathbf{t}$, ${}^{m_j}_{t}\mathbf{t}$, into constant memory, and assigning the room of global memory for POV ${}^{s}\mathbf{p}$ and wrench matrix $\mathbf{\Gamma}({}^{s}\mathbf{p})$;
2) assigning all calculating branches shown in Fig. 5 into blocksPerGrid blocks with threadsPerBlock threads contained in each block;

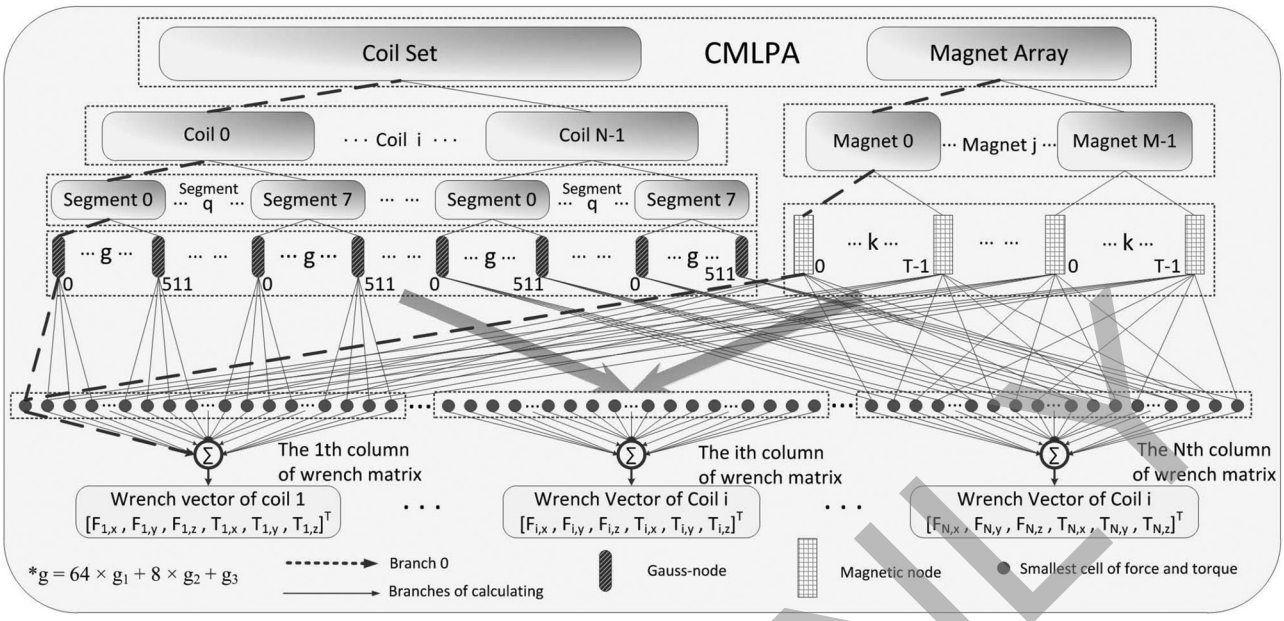

Fig. 5. Hierarchy of parallelism in PMWM.

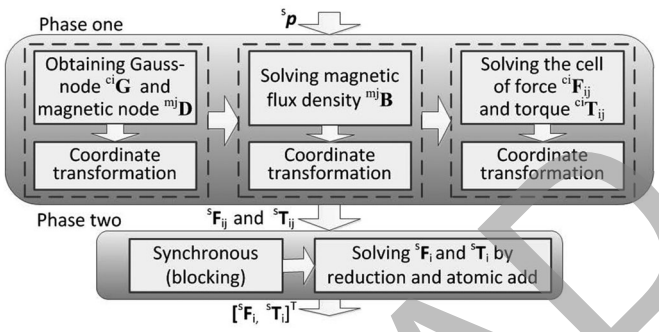

Fig. 6. Flow of each execution thread in PMWM.

3) writing the POV ${}^{s}\mathbf{p}$ into the global memory and launching the GPU kernel function;
4) reading the wrench matrix from the global memory of GPU after the kernel returns.

This is the procedure for solving the wrench matrix for a specific POV. If the wrench matrix in other POV is required, such as solving it with translator taking a specific trajectory, the main function will jump to step three and start a new loop.

2) Kernel Function on GPU Side: When the CPU launches the kernel, all of the threads will invoke the kernel function which implements the PMWM. The kernel function contains three steps as described as follows:

1) obtaining the index-array $\mathbf{v}$ with the sole ThreadID and BlockID to each thread provided by the CUDA and the weight of each variable given in Table I;
2) reading the corresponding design size of coil and magnet, rotation matrix and translation vector from the constant and global memory into the local memory;
3) solving the PMWM based on the flow showed in Fig. 6.

TABLE I
ELEMENTS OF INDEX-ARRAY $\mathbf{v}$

	Description	Range	Weight
i	Index of coil	$[0, M-1]$	$T \cdot 8^4 \cdot N$
j	Index of magnet	$[0, N-1]$	$T \cdot 8^4$
q	Coil segment	$[0, 7]$	$T \cdot 8^3$
g_1	First Gauss-node	$[0, 7]$	$T \cdot 8^2$
g_2	Second Gauss-node	$[0, 7]$	$T \cdot 8$
g_3	Third Gauss-node	$[0, 7]$	T
k	Magnetic node	$[0, T]$	1

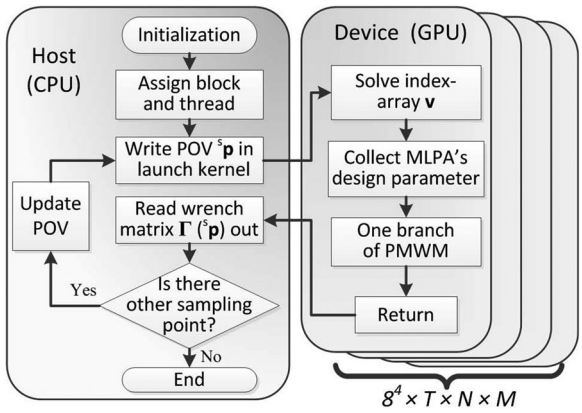

Fig. 7. Flowchart of main function and kernel function.

The pseudo code for the kernel function is given in Fig. 8. The variables in the pseudo code are aforementioned, and Γ_i means the ith column of the wrench matrix. For exp1()∼exp5(): exp1() means a mapping relationship due to (12); exp2() determines rotating matrix and translating vector; exp3() solves the Gauss-node with the node of Gaussian quadrature; exp4() and exp5()

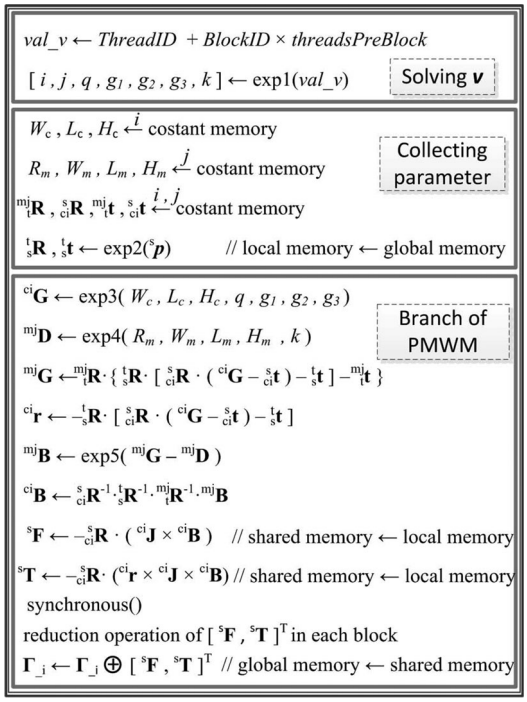

Fig. 8. Pseudo code of PMWM.

are employed to calculate the magnetic flux density based on magnetic node method. The last two steps are for obtaining the summation. The threads in one block sum up the force and torque through the reduction operator, then solve the element of wrench matrix $\mathbf{\Gamma}$ through atomic add. The symbol $\oplus$ means atomic add operation, which reads a word at some address in global memory, adds a number to it, and writes the result back to the address. The operation is atomic in the sense that it is guaranteed to be performed without interference from other threads.

D. Assignment of Threads and Memory

1) Assignment of Threads: As the amount of branches in Fig. 5 is $8^4 \times T \times N \times M$, the relationship between blocksPerGrid and threadsPerBlock meets

$$\text{blocksPerGrid} = \text{Int}\left[\frac{8^4 \times T \times N \times M - 1}{\text{threadsPerBlock}}\right] + 1 \quad (13)$$

where Int[] means a floor function that rounds the elements to the nearest integers toward minus infinity. To the GPU employed in this work, 1024 is the maximum of threadsPerBlock. With such a large number of branches in the PMWM, threadsPerBlock is set to 1024 to achieve the maximum usage of the hardware resource.

2) Assignment of Memory: The memory usage should be optimized to achieve maximum memory throughput. These global parameters, such as POV, wrench matrix, rotating matrices, and translating vectors, are accessed by all threads. Both constant memory and global memory can be read by all threads in the device (GPU) part, but unlike the global memory, the data in constant memory cannot be altered by the thread. However, constant memory is reserved for extremely low-latency read-only access to all threads; thus it is suitable to store the rotating matrices and translating vectors decided by the structure of coil set and magnet array. The POV and wrench matrix need to be defined in the global memory. With their small data size, there is not large time overhead caused by the data communication. On the other hand, each thread block has shared memory visible to all threads in the same block. Therefore, the result of force and torque solved by each thread are stored in the shared memory to perform the reduction operation in each blocks before the atomic add.

IV. Verification of the PMWM

We solve the wrench model of three kinds of CMLPAs whose topologies are shown in Fig. 9 by the PMWM. Fig. 9(a) is a structure proposed in this paper, and Fig. 9(b) and (c) are two existing designing structures proposed in [2] and [1]. In the new structure, four rectangular magnets construct a Halbach unit and the magnet array contains 16 Halbach units. These 16 units are divided in two groups to create orthogonal magnetic field in horizontal plane. The stator contains 25 independent coils. The design parameters of the CMLPA in Fig. 9(b) and (c) can be found in [2] and [1], respectively, while the design parameters of CMLPA in Fig. 9(a) are listed in Appendix.

In order to verify the calculation accuracy and efficiency, the proposed PMWM is compared with Ansys Maxwell, a universal FEM software, and Radia, a package of Mathematica for magnetostatic problem based on BEM. The same wrench model implemented on the multicore CPU, named general wrench model (GWM), accomplished parallelism through OpenMP, and it is also employed as a comparative test to show a significant acceleration of the PMWM.

In the experiment, we solve two typical issues:

1) calculating the wrench vector, which is equal to one column of wrench matrix, due to one coil in the coil set when the translator takes a certain trajectory;
2) calculating the wrench matrix of the CMPLA when the POV of translator is constant.

The Ansys Maxwell, Radia, and GWM run on CPU, and the PMWM is executed on the Kepler GPU. The specification of the hardware used for the simulation are given in Table II.

A. Wrench Vector Due to One Coil When the Translator is in a Specific Trajectory

One coil is able to represent all of the coils in the coil set because they are equivalent. We calculate the wrench vector due to the coil in red color in each coil sets in Fig. 9. The trajectory of the translator is constructed by (14). With such an equation, the translator translates and rotates simultaneously. We assume the period of simulation is 20 s

$$\begin{cases} {}^s x_p = \frac{\pi}{2} t \cdot \cos\left(\frac{\pi}{2} t\right), & {}^s \alpha_p = \frac{\pi}{2000} t, \\ {}^s y_p = \frac{\pi}{2} t \cdot \sin\left(\frac{\pi}{2} t\right), & {}^s \beta_p = \frac{\pi}{2000} t, \\ {}^s z_p = 10 + \frac{\pi}{20} t, & {}^s \gamma_p = \frac{\pi}{10} t. \end{cases} \quad (14)$$

The wrench vector of CMLPAs in Fig. 9 versus the time are shown in Fig. 10. The time steps of Radia and PMWM are both

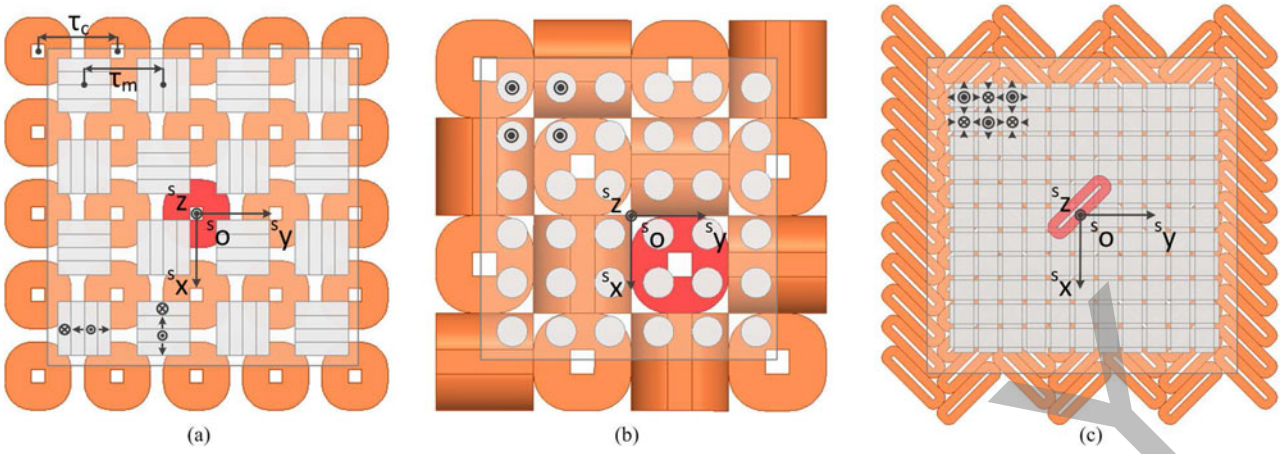

Fig. 9. Three types of CMLPA: (a) proposed structure, (b) structure in [2], (c) structure in [1].

TABLE II
HARDWARE SPECIFICATION

GPU GeForce GTX TITAN Black		CPU Intel E5-2609	
Cores	2880	Cores	8
Frequency	889 MHz	Frequency	2.4 GHz
Global Memory	4 GB	System Memory	32 GB
CUDA Version	7.0	Bus Frequency	6.4 GT/s
CUDA Capability	3.5	L3 Cache	10 MB

0.1 s. As Ansys Maxwell takes a long time to solve the force and torque at one sampling point, the time step of FEM is 1 s. Thus, there are 200 sampling points in Radia and PMWM, while 20 sampling points in FEM software.

In Table III, we list the mean error of the results solved by PMWM and Radia with respect to the solution of Ansys Maxwell to evaluate the computational accuracy. The result can be written as $[\mathbf{F},\ \mathbf{T}]^{\mathrm{T}}$, so the mean errors in force ΔF and torque ΔT are given as

$$(\Delta F\ ,\ \Delta T) = \frac{1}{\mathrm{sum}} \sum^{\mathrm{sum}} (\|\mathbf{F} - \mathbf{F}_{\mathrm{ref}}\|\ ,\ \|\mathbf{T} - \mathbf{T}_{\mathrm{ref}}\|) \qquad (15)$$

where sum, equal to 20, is the number of sampling points of Ansys Maxwell; $\mathbf{F}$ and $\mathbf{T}$ are solved by PMWM or Radia; $\mathbf{F}_{\mathrm{ref}}$ and $\mathbf{T}_{\mathrm{ref}}$ is the force and torque solve by Ansys Maxwell. Meanwhile, the average force and torque $\bar{F}$ and $\bar{T}$, given as (16), are also listed in Table III to reflect the relative error

$$(\bar{F}\ ,\ \bar{T}) = \frac{1}{\mathrm{sum}} \sum^{\mathrm{sum}} (\|\mathbf{F}_{\mathrm{ref}}\|\ ,\ \|\mathbf{T}_{\mathrm{ref}}\|). \qquad (16)$$

On the other hand, the total execution time of PMWM, GWM, and Radia are also listed in Table III to highlight the PMWM's advantage in computational efficiency.

B. Wrench Matrix Due to the Coil Set When the Translator is at a Constant Position

The wrench matrices of the CMLPAs are calculated through PMWM, Ansys Maxwell, Radia, and GWM when the POV ${}^{s}\mathbf{p}$ is $[5\ \mathrm{mm}, 5\ \mathrm{mm}, 12\ \mathrm{mm}, 0\ \mathrm{mrad}, 0\ \mathrm{mrad}, 0\ \mathrm{mrad}]$. Also using the result of Ansys Maxwell as the benchmark, error wrench matrix with respect to Radia or PMWM are defined. Fig. 11 shows the force element ΔF_x and torque element ΔT_x in the error matrices of the CMLPAs. In Fig. 11, the grid on x-axis and y-axis is integer, and represent the index of the coil in x-direction and y-direction in coil set.

To evaluate computational accuracy and efficiency, the mean error, ΔF and ΔT, and the execution time have been summarized in Table IV. At this point, sum in (15) and (16) is equal to amount of coils rather than the amount of sampling points.

C. Accuracy Analysis

Figs. 10 and 11 manifest that the force and torque solved by PMWM, Radia, and Ansys Maxwell are quite close, and the data in Tables III and IV reflect that the relative error of force and torque solved by PMWM are less than 2% and 5%, respectively. Observed from these data, the PMWM has a better accuracy than Radia if we use Ansys Maxwell as benchmark. However, in FEM software, the interaction between coil and magnet is tiny if the coil is far from the magnet. In this case, some disturbance in solving parameter, such as the meshing size or the solving region volume, is easy to affect the result of FEM. Thus, it is not rigorous to assert categorically that the accuracy of PMWM is higher than Radia, but it is positive that the accuracy of PMWM is sufficient as Radia and Ansys Maxwell.

D. Efficiency Analysis

From the execution times shown in Tables III and IV, it is obvious that the PMWM has a significant improvement in computational efficiency compared with the other two methods. Since there are 200 times data communication in the test 1, the speedup ratio of the first test is lower than that of the second one. Thus, we cannot ignore the time overhead of data communication between host and device if we want to develop the computational efficiency of PMWM further.

Fig. 12 shows the execution time of PMWM on GPU and GWM on CPU versus the different scale of CMLPAs. Benefited

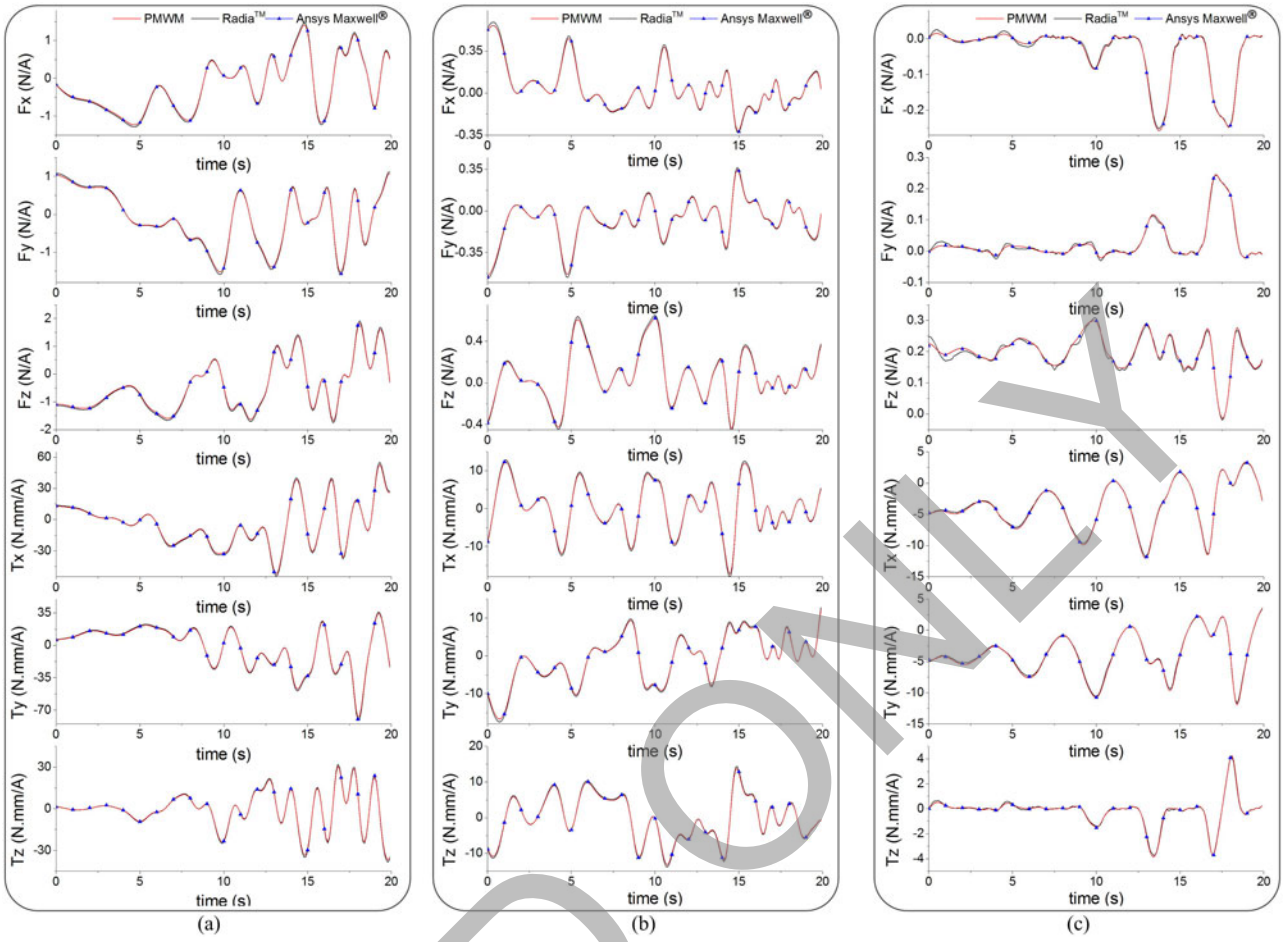

Fig. 10. Forces and torques acting on the magnet array due to a single coil in the CMLPAs. (a), (b), and (c) correspond to the three CMLPAs in Fig. 9, respectively.

TABLE III
COMPARISON OF COMPUTATION ACCURACY AND EFFICIENCY BETWEEN PMWM, RADIA, ANSYS MAXWELL, AND GWM IN TEST 1

CMLPA	Computation accuracy						Computation efficiency				
	ΔF (mN/A)		$\bar{F}$ (mN/A)	ΔT (N.mm/A)		$\bar{T}$ (N.mm/A)	Execution time (s)			Speed up	
	PMWM	Radia		PMWM	Radia		PMWM	Radia	GWM	Radia	GWM
Fig. 9(a)	4.9718	6.7954	328.2719	0.1960	0.2185	7.7215	1.091	17.457	33.962	16.001	31.129
Fig. 9(b)	1.2359	2.4270	51.6644	0.0305	0.0401	1.5899	6.697	115.05	308.51	17.179	46.067
Fig. 9(c)	1.6211	2.5795	84.2910	0.0497	0.0601	2.3856	4.573	71.366	188.77	15.606	41.279

from the high parallelism of the proposed massively parallel model and the massive number of cores in the GPU, the execution time of PMWM on GPU is much less and more linear than that of GWM on CPU.

A vital reason why the ratio between execution time and threads amount is linear on GPU is that the PMWM eliminates the data exchange between different thread blocks. As threadsPerBlock equals to 1024 aforementioned, the computation efficiency of the PMWM is benefited from choosing an integer power of 2 number branches. Therefore, we chose the eight-order Gaussian quadrature and eight segments for the coil. In this case, the threads in the same block are for the wrench vector of the same coil. The reduction operation executes in the same blocks rather than access the shared memory in different block, so that the uncertain time overhead is reduced. This result also proves that the PMWM uses the GPU adequately, and distributes the hardware resource reasonably.

E. Comparison Between the Existing Calculating Method

The analytical wrench models for the other two designs in Fig. 9 are proposed in previous papers, such as the magnetic

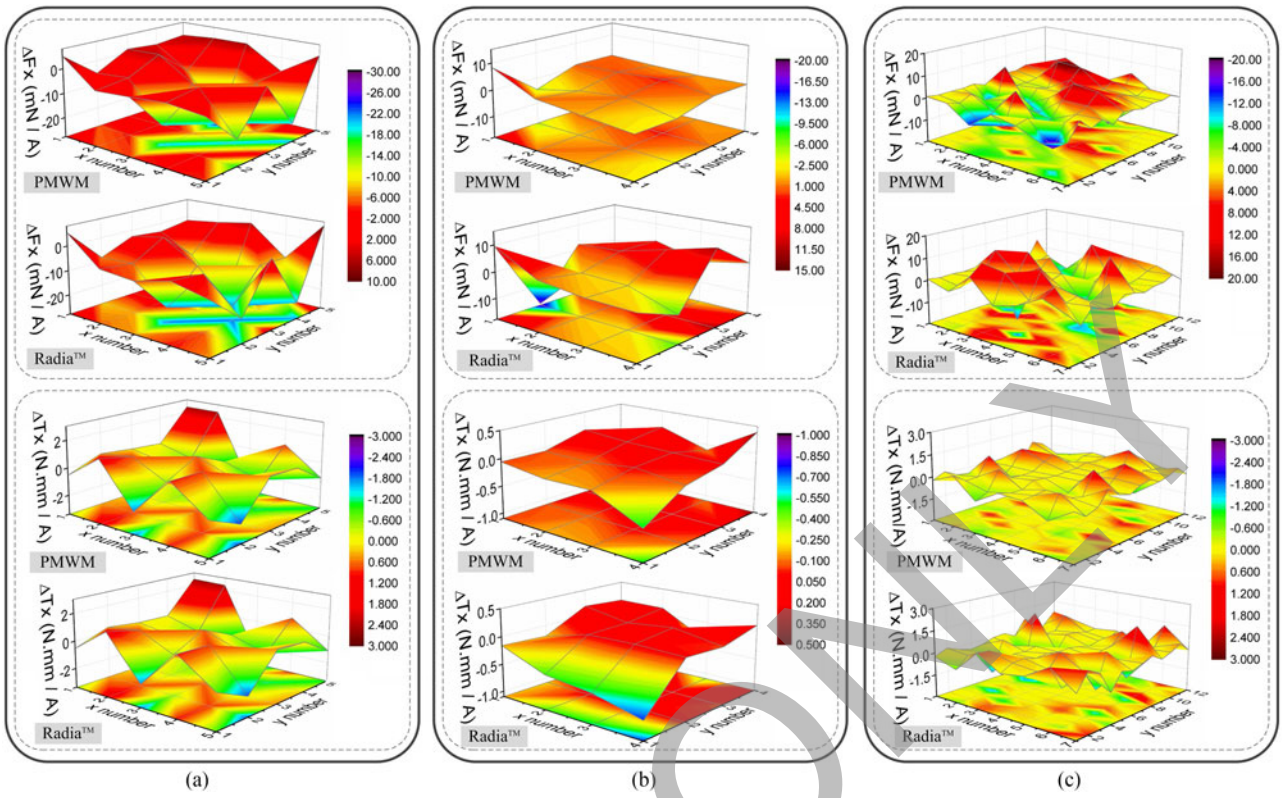

Fig. 11. Elements ΔF_x and ΔT_x in the error matrices of the CMPLAs. (a), (b), and (c) correspond to the three CMLPAs in Fig. 9, respectively. Top figures are the differences between Ansys Maxwell and PMWM, while the under figures are the differences between Ansys Maxwell and Radia.

TABLE IV
COMPARISON OF COMPUTATION ACCURACY AND EFFICIENCY BETWEEN PMWM, RADIA, ANSYS MAXWELL, AND GWM IN TEST 2

CMLPA	Computation accuracy						Computation efficiency				
	ΔF (mN/A)		$\bar{F}$ (mN/A)	ΔT (N.mm/A)		$\bar{T}$ (N.mm/A)	Execution time (s)			Speed up	
	PMWM	Radia		PMWM	Radia		PMWM	Radia	GWM	Radia	GWM
Fig. 9(a)	3.2706	4.0101	189.8504	0.3782	0.3808	15.3386	0.0901	2.0851	3.979	23.142	44.162
Fig. 9(b)	1.2672	3.0966	47.0408	0.0556	0.1015	1.2651	0.5439	11.081	29.042	20.373	53.396
Fig. 9(c)	0.8223	1.2953	45.3455	0.0939	0.1159	4.5334	1.8611	39.341	61.595	21.138	33.096

charge method is employed in [2], while the harmonic method and the magnetic charge method are employed in [44].

The experiments in these works showed that the results of magnetic charge method are effective and close to the testing. Since the magnetic charge method is computationally expensive, these works simplify the model of coil or magnet to decrease the computational burden. In [2], the cylindrical magnet in the translator is equivalent to two points coincident with central point of top and bottom surfaces, respectively, and in [44], only four surfaces in the corners of the coil are meshed. So these methods will produce significant error when the translator takes large rotational motion. Compared with these methods, the PMWM keeps the specific of design structure, and is available in any case of motion with similar execution time.

The result of the harmonic method employed in [44] is good, but also shows the unavoidable end-effect and restriction in the transnational DOFs. In this method, the force and torque can be calculated for most part analytically, so the calculation time is very fast. However, as the current carrying region and permanent magnets should be spaced periodical to reduce the effect of high harmonics, the application scope of harmonic method is limited; the other CMLPAs, such as the designs of Fig. 9(a) and (b) cannot use this method. Compared with the harmonic method, the PWMM has the advantage of the generality and robustness.

However, there are some limitations of this PMWM. Since the PMWM is not a time-domain model, it is only suitable for the CMPLA with direct dc driving, but not as effective to solve wrench model of the ac planar motors as the harmonic method. On the other hand, the iron in the planar motor produces plenty of virtual magnetic nodes, which results in larger computational burden. To the iron planar motor, the equivalent magnetic circuit method or harmonic method can be a suitable alternative.

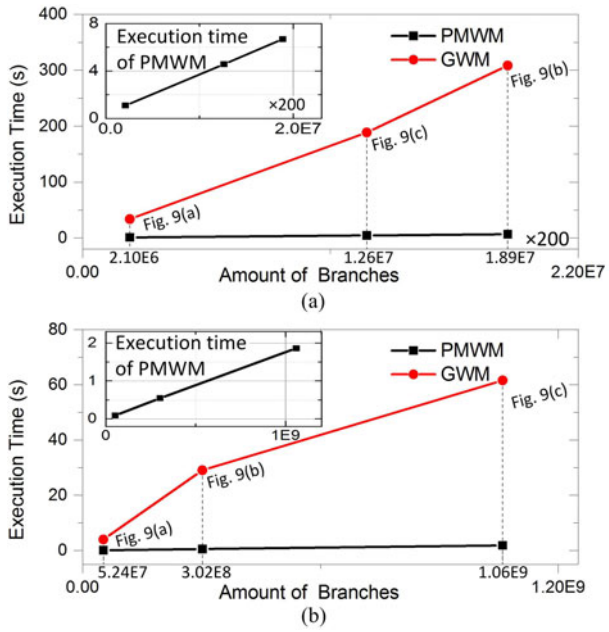

Fig. 12. Execution time of PMWM on GPU and GWM on CPU versus different amount of branches in the parallel wrench model for the three CMLPAs in (a) test 1 and (b) test 2.

V. Conclusion

An accurate and efficient wrench model for the CMPLA based on magnetic node, Gaussian quadrature, and coordinate transformation, was proposed. The PMWM was implemented on the GPU to take advantage of massive parallelism. Several tests were undertaken to evaluate the performance of the parallel model. The FEM software Ansys Maxwell was used as benchmark to verify the computational accuracy, while the BEM software Radia and the GWM implemented on CPU through OpenMP were used to highlight the improvement in execution efficiency. The results show that the PMWM achieves significant computation speedup with sufficient accuracy.

Appendix

The design parameters of the proposed CMLPA are given as follow: $\tau_c = 60$ mm, $\tau_m = 60$ mm, $R_{\text{cin}} = 0$ mm, $R_{\text{cout}} = 20$ mm, $L_c = 5$ mm, $W_c = 5$ mm, $H_c = 15$ mm, $L_m = 20$ mm, $W_m = 5$ mm, $H_m = 5$ mm, the remanence is 1T, and the turns of the coil is 1000.

References

[1] J. M. M. Rovers, J. W. Jansen, J. C. Compter, and E. A. Lomonova, "Analysis method of the dynamic force and torque distribution in the magnet array of a commutated magnetically levitated planar actuator," *IEEE Trans. Ind. Electron.*, vol. 59, no. 5, pp. 2157–2166, May 2012.

[2] K. S. Kung and Y. S. Beak, "Study on a novel contact-free planar system using direct drive DC coils and permanent magnets," *IEEE/ASME Trans. Mechatronics*, vol. 7, no. 7, pp. 35–43, Mar. 2002.

[3] M. B. Binnard, "Planar motor with linear coil arrays," U.S. Patent 6 445 093, Sep. 3, 2002.

[4] J. D. Boeij, E. A. Lomonova, and J. Duarte, "Contactless planar actuator with manipulator: A motion system without cables and physical contact between the mover and the fixed world," *IEEE Trans. Ind. Appl.*, vol. 45, no. 6, pp. 1930–1938, Nov./Dec. 2009.

[5] S. Xiao and Y. Li, "Optimal design, fabrication, and control of an XY micropositioning stage driven by electromagnetic actuators," *IEEE Trans. Ind. Electron.*, vol. 60, no. 10, pp. 4613–4626, Oct. 2013.

[6] Y. Zhu, C. Hu, J. Hu, and K. Yang, "Accuracy- and simplicity-oriented self-calibration approach for two-dimensional precision stages," *IEEE Trans. Ind. Electron.*, vol. 60, no. 6, pp. 2264–2272, Jun. 2013.

[7] M. Sitti, "Survey of nanomanipulation systems," in *Proc. 1st IEEE Conf. Nanotechnol.* 2001, pp. 75–80.

[8] M. B. Binnard, "System and method to control planar motors," U.S. Patent 6 650 079, Nov. 18, 2003.

[9] C. M. M. van Lierop, J. W. Jansen, E. A. Lomonova, A. A. H. Damen, P. P. J. van den Bosch, and A. J. A. Vandenput, "Commutation of a magnetically levitated planar actuator with moving-magnets," *IEEJ Trans. Ind. Appl.*, vol. 128, no. 12, pp. 1333–1338, Dec. 2008.

[10] L. Petit, A. Hassine, J. Terrien, F. Lamarque, and C. Prelle, "Development of a control module for a digital electromagnetic actuators array," *IEEE Trans. Ind. Electron.*, vol. 61, no. 9, pp. 4788–4796, Sep. 2014.

[11] M. Y. Chen, T. B. Lin, S. K. Huang, and L. C. Fu, "Design and experiment of a macro-micro planar maglev positioning system," *IEEE Trans. Ind. Electron.*, vol. 59, no. 11, pp. 4128–4139, Nov. 2012.

[12] J. P. C. Smeets, T. T. Overboom, J. W. Jansen, and E. A. Lomonova, "Modeling framework for contactless energy transfer systems for linear actuators," *IEEE Trans. Ind. Electron.*, vol. 60, no. 1, pp. 391–399, Jan. 2013.

[13] W. J. Kim and D. L. Trumper, "High-precision magnetic levitation stage for photolithography," *Precision Eng.*, vol. 22, no. 2, pp. 66–77, Apr. 1998.

[14] T. J. Teo, H. Zhu, and C. K. Pang, "Modeling of a two degrees-of-freedom moving magnet linear motor for magnetically levitated positioners," *IEEE Trans. Magn.*, vol. 50, no. 12, Dec. 2014, Art. no. 8300512.

[15] J. W. Jansen, J. P. C. Smeets, T. T. Overboom, J. M. M. Rovers, and E. A. Lomonova, "Overview of analytical models for the design of linear and planar motors," *IEEE Trans. Magn.*, vol. 50, no. 11, Nov. 2014, Art. no. 8208207.

[16] W. J. Kim, S. Verma, and H. Shakir, "Design and precision construction of novel magnetic-levitation-based multi-axis nanoscale positioning systems," *Precis. Eng.*, vol. 31, no. 4, pp. 337–350, Oct. 2007.

[17] M. Y. Chen, H. H. Huang, and S. K. Hung, "A new design of a submicropositioner utilizing electromagnetic actuators and flexure mechanism," *IEEE Trans. Ind. Electron.* vol. 57, no. 1, pp. 96–106, Jan. 2010.

[18] H. Zhang, B. Kou, H. Zhang, and Y. Jin, "A three-degree-of-freedom short-stroke Lorentz-force-driven planar motor using a Halbach permanent-magnet array with unequal thickness," *IEEE Trans. Ind. Electron.*, vol. 62, no. 6, pp. 3640–3650, Jun. 2015.

[19] F. Xu, X. Xu, Z. Li, and L. Chu, "Numerical calculation of the magnetic field and force in cylindrical single-axis actuator," *IEEE Trans. Magn.*, vol. 50, no. 10, Oct. 2014, Art. no. 7200506.

[20] M. Y. Chen, M. J. Wang, and L. C. Fu, "Modeling and controller design of a maglev guiding system for application in precision positioning," *IEEE Trans. Ind. Electron.* vol. 50, no. 3, pp. 493–506, Jun. 2003.

[21] A. El Hajjaji and M. Ouladsine, "Modeling and nonlinear control of magnetic levitation systems," *IEEE Trans. Ind. Electron.*, vol. 48, no. 4, pp. 831–838, Aug. 2001.

[22] D. L. Trumper, S. M. Olson, and P. K. Subrahmanyan, "Linearizing control of magnetic suspension systems," *IEEE Trans. Control Syst. Technol.*, vol. 5, no. 4, pp. 427–438, Jul. 1997.

[23] C. Pompermaier, K. Kalluf, A. Zambonetti, M. V. Ferreira da Luz, and I. Boldea, "Small linear PM oscillatory motor: Magnetic circuit modeling corrected by axisymmetric 2-D FEM and experimental characterization," *IEEE Trans. Ind. Electron.*, vol. 59, no. 3, pp. 1389–1396, Mar. 2012.

[24] J. Wang, D. Howe, and Z. Lin, "Design optimization of short-stroke single-phase tubular permanent-magnet motor for refrigeration applications," *IEEE Trans. Ind. Electron.*, vol. 57, no. 1, pp. 327–334, Dec. 2009.

[25] L. Yu and T. N. Chang, "Zero vibration on-off position control of dual solenoid actuator," *IEEE Trans. Ind. Electron.*, vol. 57, no. 7, pp. 2519–2526, Jul. 2010.

[26] P. Estevez, A. Mulder, and R. M. Schmidt, "6-DoF miniature maglev positioning stage for application in haptic micro-manipulation," *Mechatronics*, vol. 22, no. 7, pp. 1015–1022, Oct. 2012.

[27] T. D. Nguyen, K. J. Tseng, S. Zhang, and H. T. Nguyen, "A novel axial flux permanent-magnet machine for flywheel energy storage system: Design and analysis," *IEEE Trans. Ind. Electron.*, vol. 58, no. 9, pp. 3784–3794, Sep. 2011.

[28] P. Berkelman and M. Dzadovsky, "Magnetic levitation over large translation and rotation ranges in all directions," *IEEE/ASME Trans. Mechatronics*, vol. 18, no. 1, pp. 44–52, Feb. 2013.

[29] O. Chubar, P. Elleaume, and J. Chavanne, "A three-dimensional magnetostatics computer code for insertion devices," *J. Synchrotron Radiation*, vol. 5, pp. 481–484, May 1998.
[30] Z. Andjelic, G. Of, O. Steinbach, and P. Urthaler, "Boundary element methods for magnetostatic field problems: A critical view," *Comput. Vis. Sci.*, vol. 14, no. 3, pp. 117–130, Mar. 2011.
[31] R. Chang, S. Li, M. V. Lubarda, B. Livshitz, and V. Lomakin, "FastMag: Fast micromagnetic simulator for complex magnetic structures," *J. Appl. Phys.*, vol. 109, no. 7, Apr. 2011, Art. no. 07D358.
[32] T. Okimura, T. Sasayama, N. Takahashi, and S. Ikuno, "Parallelization of finite element analysis of nonlinear magnetic fields using GPU," *IEEE Trans. Magn.*, vol. 49, no. 5, pp. 1557–1560, May 2013.
[33] F. Bancel, "Magnetic nodes," *J. Phys. D, Appl. Phys.*, vol. 32, no. 17, pp. 2155–2161, Mar. 1999.
[34] D. Kahaner, C. Moler, and S. Nash, *Numerical Methods and Software*, Englewood Cliffs, NJ, USA: Prentice-Hall, 1989.
[35] J. D. Owens, M. Houston, D. Luebke, S. Green, J. E. Stone, and J. C. Phillips, "GPU computing," *Proc. IEEE*, vol. 96, no. 5, pp. 879–899, May 2008.
[36] L. Leroy, P. Fuchs, and G. Moreau, "Visual fatigue reduction for immersive stereoscopic displays by disparity, content, and focus-point adapted blur," *IEEE Trans. Ind. Electron.*, vol. 59, no. 10, pp. 3998–4004, Oct. 2012.
[37] H. Wu, L. Lou, C. Chen, S. Hirche, and K. Kuhnlenz, "Cloud-based networked visual servo control," *IEEE Trans. Ind. Electron.*, vol. 60, no. 2, pp. 554–566, Feb. 2013.
[38] V. Jalili-Marandi and V. Dinavahi, "SIMD-based large-scale transient stability simulation on the graphics processing unit," *IEEE Trans. Power Syst.*, vol. 25, no. 3, pp. 1589–1599, Aug. 2010.
[39] V. Jalili-Marandi, Z. Zhou, and V. Dinavahi, "Large-scale transient stability simulation of electrical power systems on parallel GPUs," *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, no. 7, pp. 1255–1266, Jul. 2012.
[40] Z. Zhou and V. Dinavahi, "Parallel massive-thread electromagnetic transient simulation on GPU," *IEEE Trans. Power Del.*, vol. 29, no. 3, pp. 1045–1054, Jun. 2014.
[41] Z. Liu, X. Li, L. Wu, S. Zhou, and K. Liu, "GPU-accelerated parallel coevolutionary algorithm for parameters identification and temperature monitoring in permanent magnet synchronous machines," *IEEE Trans. Ind. Informat.*, vol. 11, no. 5, pp. 1220–1230, Oct. 2015.
[42] H. Karimipour and V. Dinavahi, "Extended Kalman filter based parallel dynamic state estimation," *IEEE Trans. Smart Grid*, vol. 6, no. 3, pp. 1539–1549, May 2015.
[43] H. Karimipour and V. Dinavahi, "Parallel relaxation-based joint dynamic state estimation of large-scale power systems," *IET Gener., Transmiss. Distrib.*, vol. 10, no. 2, pp. 452–459, Feb. 2016.
[44] J. W. Jansen, "Magnetically levitated planar actuator with moving magnets: Electromechanical analysis and design," Ph.D. dissertation, Dept. Elect. Eng., Eindhoven Univ. Technol., Eindhoven, The Netherlands, 2007.

Fengqiu Xu (S'15) was born in Fengcheng, China. He received the B.Eng. degree in measurement and control technology and instruments from Wuhan University, Wuhan, China, in 2011. He is currently working toward the joint Ph.D. degree in the Electrical Information School, Wuhan University, and the University of Alberta, Edmonton, AB, USA.

His research interests include magnetically levitated technology and real-time simulation of electrical machines.

Venkata Dinavahi (SM'08) received the Ph.D. degree from the University of Toronto, Toronto, ON, Canada, in 2000.

He is currently a Professor of electrical and computer engineering at the University of Alberta, Edmonton, AB, Canada. His research interests include real-time simulation of electrical machines, power electronics and power systems, large-scale system simulation, and parallel and distributed computing.

Xianze Xu received the B.Eng. degree in mechanical engineering from Hefei University of Technology, Hefei, China, in 1990, and the M.Eng. and Ph.D. degrees in the same area from Wuhan University of Technology, Wuhan, China, in 1992 and 2002, respectively.

He is currently a Professor in the Electrical Information School, Wuhan University, Wuhan, China. His research interests include precision instruments design, manufacture, and control.