Sparsity-Aware Precorrected Tensor Train Algorithm for Fast Solution of 2-D Scattering Problems and Current Flow Modeling on Unstructured Meshes

Zhuotong Chen, *Student Member, IEEE*, Luis J. Gomez, *Member, IEEE*, Shucheng Zheng, *Student Member, IEEE*, Abdulkadir C. Yucel, *Senior Member, IEEE*, Zheng Zhang, *Member, IEEE*, and Vladimir I. Okhmatovski, *Senior Member, IEEE*

***Abstract*— Acceleration of the method of moments (MoM) solution of the volume integral equation (VIE) on unstructured meshes is performed using a precorrected tensor train (P-TT) algorithm. The elements of the MoM's unstructured mesh are projected onto a regular Cartesian grid. This enables representation of the MoM matrix as the Toeplitz matrix of point-to-point interactions pre- and post-multiplied by sparse matrices projecting MoM's basis and testing functions on the Cartesian grid. The Toeplitz matrix is subsequently cast into the form of a multidimensional tensor. The latter is decomposed into the product of smaller dimensional matrices also known as tensor train (TT). TT allows to store Toeplitz matrix in $O(\log N)$ memory for VIE with the Laplace kernel and in $O(N \log N)$ memory for VIE with the Helmholtz kernel. Unlike the FFT-based fast algorithms, the P-TT method enables further memory reduction due to the sparsity of sources on the Cartesian grid. This sparsity pattern can be accounted for in construction of TT cores. It further reduces memory use as well as CPU time required for the multiplication of the Toeplitz matrix with a vector. It is shown that upon sufficient sparsity in representation of the sources on the grid, the P-TT algorithm can outperform both in CPU time and memory its FFT-based counterpart known as the precorrected FFT algorithm.**

***Index Terms*— Fast algorithms, scattering, tensor train (TT) decomposition.**

Manuscript received May 4, 2019; revised August 10, 2019; accepted October 6, 2019. Date of publication November 26, 2019; date of current version December 27, 2019. This work was supported by the Collaborative Research and Development Grant from the Natural Sciences and Engineering Research Council (NSERC) and Manitoba Hydro International. *(Corresponding author: Vladimir I. Okhmatovski.)*

Z. Chen and Z. Zhang are with the Department of Electrical and Computer Engineering, University of California at Santa Barbara, Santa Barbara, CA 93106 USA (e-mail: ztchen@ucsb.edu; zhengzhang@ece.ucsb.edu).

L. J. Gomez is with the Psychiatry and Behavioral Sciences Department, Duke University School of Medicine, Durham, NC 27710 USA (e-mail: ljg24@duke.edu).

S. Zheng and V. I. Okhmatovski are with the Department of Electrical and Computer Engineering, University of Manitoba, Winnipeg, MB R3T 5V6, Canada (e-mail: umzheng6@myumanitoba.ca; vladimir.okhmatovski@umanitoba.ca).

A. C. Yucel is with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore (e-mail: acyucel@ntu.edu.sg).

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

Digital Object Identifier 10.1109/TMTT.2019.2948873

I. Introduction

FAST algorithms of computational electromagnetics (CEM) have become essential in the design cycles of complex microwave circuits, analysis of electrically large photonic and optical devices, solution of radiation problems for vehicle-mounted antennas, and many other important areas. These algorithms allow drastic reduction of computational time and memory in the solution of boundary value problems of electromagnetics through the use of efficient data structures and optimal representation of the matrix operators resulting from discretization of the pertinent integral or differential equations. Fast algorithms of CEM allow to perform analysis of larger and/or more complex structures, more accurately predict the scattering and radiation phenomena, and, ultimately, enable better designs of various devices and systems.

First debut of the fast algorithms into CEM was made in 1971 [1] by what later became known as the conjugate-gradient fast Fourier transform (CG-FFT) algorithm. The method used FFT for the acceleration of scattered field computations produced by given volumetric currents, which were cast into the form of convolutions through the Method of Moments (MoM) discretization of the volume integral equation (VIE) on the regular rectangular grids. The CG-FFT method, however, remained largely unnoticed until mid-1980s [2] when static tree-based algorithms, such as the Appel algorithm [3], Barnes–Hut algorithm [4], and fast multiplole method (FMM) [5], were introduced into computational physics at large and CEM in particular. While the FMM became the method of choice for the solution of large-scale radiation and scattering problems by mid-90s due to its generalization to full-wave kernel [6] and three dimensions [7], the FFT methods remained popular due to the simplicity of FFT use, its efficiency, suitability of FFT acceleration schemes to solution of problems in complex media [8], [9], and their generalization to handle discretizations on unstructured meshes [10]–[13]. The FFT methods and FMM have dominated the landscape of fast algorithms in CEM throughout the 90s and most of 2000s until the theory and applications of hierarchical matrices ($\mathcal{H}$-matrices) had matured to offer principally new capabilities of fast direct solution for both dense and sparse matrix equations [14]. Iterative nature of the FFT and FMM methods made their application to solutions of the multiscale problems and/or problems with large disparity in material properties difficult and often impossible. The poor conditioning of the pertinent matrix equations prevents iterative matrix solvers [15], [16] from converging to a solution, and construction of effective preconditioners becomes as computationally expensive as the direct naive solution of the original matrix equation. The ability of the $\mathcal{H}$-matrix-based methods to solve ill-conditioned equations directly and with comparable use of computational resources to those of iterative methods made them widely popular for the solution of problems defying FMM- and FFT-based iterative solvers [17], [18].

Tensor train (TT) decomposition of the Toeplitz matrices introduced by Oseledets [19] in 2009 took reduction of computational resources required for the solution of boundary problems of CEM to a principally new level. While FFT-based algorithms required $O(N \log N)$ memory to store Toeplitz matrices, their decomposition into a product of smaller dimensional matrices could be performed in unprecedented $O(\log N)$ operations and stored in $O(\log N)$ memory for the problems of electrically moderate sizes. Such drastic reduction in operations required to factor the Toeplitz matrix and memory needed for its storage may result from the compression of the generating matrix, i.e., the matrix of unique elements in the Toeplitz matrix. The latter is computed in $O(N)$ operations and stored in $O(N)$ memory in the FFT-based fast methods. The TT-decomposed $N \times N$ matrix occupying $O(\log N)$ storage can also be directly inverted in $O(\log N)$ operations [20], [21] and applied to the N element vector of excitation provided the latter allows for $O(\log N)$ representation and so does the solution of the matrix equation. In case of MoM discretized VIE and its Toeplitz matrices, such efficient solutions of the TT-decomposed matrix equation are currently only available for discretizations of exactly rectangular geometries with exactly rectangular grids.

The TT-decomposed matrix can also be multiplied with an arbitrary vector of length N and used for construction of iterative fast solvers. However, the matrix-vector-product (MVM) of TT is performed with an arbitrary vector in $O(N \log N)$ operations. When it comes to MoM solution of the VIE, it also requires that MoM discretization to be performed on ideal square grids. The object must also be voxelized. In our previous work [22], we accelerated MoM solution of VIE utilizing square grids by decomposing its pertinent 4-D Toeplitz matrix into TT and performing MVMs with it within iterative matrix equation solvers, such as the conjugate-gradient (CG) method [15] and others [16]. We called developed fast algorithm "CG-TT" [22] by analogy with the popular CG-FFT method [8], [23], [24] as it has all the attributes of the latter while offering additional memory savings stemming from $O(\log N)$ Toeplitz matrix representation in TT format and utilization of sparsity in the occupancy of the grid by the sources.

In order to decouple the MoM discretization from the square-grid representation of the currents and scattered fields required by TT decomposition, we subsequently introduced the concept of precorrection used in FFT-based fast algorithms, such as the precorrected FFT (P-FFT) [10], [11] and the adaptive integral method (AIM) [12], [13], to effect such decoupling into construction of a TT-based fast iterative solver [25]. We termed this approach the precorrected TT (P-TT) algorithm due to the similarity of its attributes to the well-known P-FFT algorithm [10]–[13].

Both CG-TT [22] and P-TT [25] fast iterative solvers were shown to easily outperform CG-FFT [2] and P-FFT [11] algorithms in terms of memory usage. However, they exhibited notably inferior performance compared to the FFT-based algorithms in terms of CPU time required for computations of the MVMs. This inferior CPU time performance of TT compared to FFT acceleration is partially due to the maturity of the latter and novelty of the former. However, despite highly optimized implementation of FFTs, evaluation of MVM with the TT-decomposed Toeplitz matrix can be made more efficient if the geometry of interest exhibits substantial sparsity. Such sparsity is commonly observed in the case of surface integral equation solutions with rectangular-grid-methods (FFT- or TT-based) when only the grid point near the surface of the object attains nonzero current values. Substantial sparsity in the representation of the currents on the regular grids also occurs in the VIE solutions when scatterers are well separated. The FFT-based methods cannot exploit this sparsity of the grid and are forced to perform operation on the zero-valued currents. This is the reason for their $O(N^{1.5} \log N)$ scaling in the case of 3-D surface integral equation solutions. The TT decomposition, however, can take advantage of such sparsity. The higher is the sparsity in the object's representation on the regular grid, the more drastic is the reduction in time and memory required for performing multiplication of the tensor cores in the TT with a vector. Contribution of this article and its nontrivial extension compared to the original P-TT algorithm introduced in [25] is in the utilization of sparsity to increase the performance of the TT decomposition and computation of MVMs with TT-decomposed Toeplitz matrices. In order to enable sparsity-aware TT core construction, the TT decomposition of the Toeplitz matrix is done at the level of the binary trees corresponding to the partitioned source and observation domains as opposed to the TT decomposition in [22] and [25] performed according to the quad-tree representing the 4-D Toeplitz matrix itself. The advantages of the sparsity-aware P-TT algorithm compared to the classical P-FFT [11] fast algorithm are demonstrated via the solution of both full-wave 2-D scattering problems and magneto-quasi-static problems of inductance extractions in 2-D multiconductor transmission lines (MTLs).

II. Voxilization of MoM on Unstructured Meshes for VIE

The magneto-quasi-statics VIE in 2-D case is

$$E_z(\boldsymbol{\rho}) + \imath\omega\mu_0\sigma \int_S \mathbb{G}(\boldsymbol{\rho}, \boldsymbol{\rho}')E_z(\boldsymbol{\rho}')ds' = V^{\text{p.u.l.}}, \quad \boldsymbol{\rho} \in S \qquad (1)$$

where σ is the conductivity of wire, μ_0 is the vacuum permeability, $\mathbb{G}(\boldsymbol{\rho}, \boldsymbol{\rho}') = g(|\boldsymbol{\rho} - \boldsymbol{\rho}'|) = -1/(2\pi)\ln(|\boldsymbol{\rho} - \boldsymbol{\rho}'|)$ is the 2-D free space Green's function, and $V^{\text{p.u.l}}$ is the voltage drop along the wire, $\imath$ being $\sqrt{-1}$.

To solve VIE numerically, the MoM expands the unknown current density of conductivity $\mathtt{j}_z = \sigma E_z$ over N pulse basis functions b_s as

$$\mathtt{j}_z(\boldsymbol{\rho}) = \sum_{s=1}^{N} \mathtt{j}_s b_s(\boldsymbol{\rho}) \tag{2}$$

and tests the resulting equation with the same set of the test function. This produces the following SLAE $\mathbf{Z}\cdot\mathbf{j} = \mathbf{v}^{\text{p.u.l.}}$:

$$(\sigma^{-1}\mathbf{I} + i\omega\mathbf{L})\cdot\mathbf{j} = \mathbf{v}^{\text{p.u.l.}} \tag{3}$$

where

$$L_{s,s'} = \langle b_s, \langle \mathtt{G}, b_{s'}\rangle\rangle = \mu_0 \int_{S_s}\int_{S_{s'}} \mathtt{G}(\boldsymbol{\rho},\boldsymbol{\rho}')ds'ds \tag{4}$$

where $s, s' = 1, \ldots, N$, N is the number of triangles discretizing the cross section of the transmission line, $\mathbf{I}$ is the idemfactor, $S_{s'}$ and S_s are the areas of the source and observation triangles, respectively, and $\mathbf{j} = [\mathtt{j}_1, \mathtt{j}_2, \ldots, \mathtt{j}_N]$ is the vector of unknown coefficients in current expansion.

In order to TT accelerate the MVM in the fast iterative solution of matrix equation in unstructured mesh MoM, the interactions of N source triangles with each observation triangle are partitioned into near and far analogously with the P-FFT algorithm [10], yielding SLAE (3) in the form

$$(\sigma^{-1}\mathbf{I} + \imath\omega\mathbf{L}^{\text{near}} + \imath\omega\mathbf{L}^{\text{far}})\cdot\mathbf{j} = \mathbf{v}^{\text{p.u.l.}} \tag{5}$$

where $\mathbf{L}^{\text{near}}$ is the matrix of MoM near interactions, i.e., the matrix of MoM test basis functions interacting with their near neighbors, and computed directly according to (4). For a given observation test function b_s, the near interacting source functions $b_{s'}$ are partially or fully situated within a circle of given radius R^{near} centered at the centroid of triangle supporting b_s.

The matrix $\mathbf{L}^{\text{far}}$ in (5) is the matrix of MoM far interactions, such that $\mathbf{L} = \mathbf{L}^{\text{near}} + \mathbf{L}^{\text{far}}$. Each element $L_{s,s'}$ of matrix $\mathbf{L}^{\text{far}}$ can be approximated with controlled precision if the original basis and test functions $b_{s'}$ and b_s are replaced with their equivalent representations $\tilde{b}_{s'}$ and $\tilde{b}_s$ formed by the point sources snapped to the nodes of the regular $(Q_x \times Q_y)$ grid as

$$\tilde{b}_s(\boldsymbol{\rho}) = \sum_{q'_x=1}^{Q_x}\sum_{q'_y=1}^{Q_y} \Lambda^s_{q'_x q'_y}\delta(\boldsymbol{\rho} - \boldsymbol{\rho}_{q'_x q'_y}). \tag{6}$$

In (6), only Q^2 point source coefficients out of $Q_x \times Q_y$ are nonzero.

They can be determined using so-called multipole-reproduction-criteria [13] producing the following Vandermonde system of linear equations:

$$\sum_{q_1=1}^{Q}\sum_{q_2=1}^{Q} \Lambda^s_{q_1q_2}\left(x^s_{q_1q_2} - x^s_0\right)^{\mu}\left(y^s_{q_1q_2} - y^s_0\right)^{\nu} = \int_{S_s}\left(x - x^s_0\right)^{\mu}\left(y - y^s_0\right)^{\nu} dxdy. \tag{7}$$

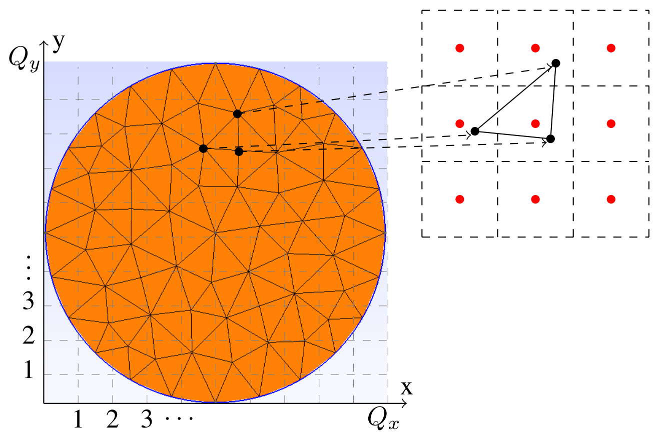


Fig. 1. Stencil of nine point sources $Q^2 = 9$ corresponding to the projection of the highlighted triangle mesh element onto the regular Cartesian grid. The center point of the stencil is the closest grid point to the centroid of its corresponding triangle element. The magnitudes of the points sources are calculated in the same way as it is done in the P-FFT method [10].

The elements $L_{s,s'}$ of the matrix of far interactions $\mathbf{L}^{\text{far}}$ in (5) are approximated with matrix elements

$$\begin{aligned}\mathcal{L}_{s,s'} &= \langle \tilde{b}_s, \langle \mathtt{G}, \tilde{b}_{s'}\rangle\rangle \\ &= \mu_0 \int_{S_s} \tilde{b}_s(\boldsymbol{\rho}) \int_{S_{s'}} \mathtt{G}(\boldsymbol{\rho},\boldsymbol{\rho}')\tilde{b}_{s'}(\boldsymbol{\rho}')ds'ds \\ &= \sum_{q'_x=1}^{Q_x}\sum_{q'_y=1}^{Q_y}\sum_{q_x=1}^{Q_x}\sum_{q_y=1}^{Q_y} \Lambda^s_{q_xq_y}\mathtt{G}(\boldsymbol{\rho}_{q_xq_y}, \boldsymbol{\rho}_{q'_xq'_y})\Lambda^{s'}_{q'_xq'_y}\end{aligned} \tag{8}$$

where $\tilde{b}_s$ and $\tilde{b}_{s'}$ are sth test and s'th basis obtained by replacing original test and basis functions b_s and $b_{s'}$, respectively, with Q^2 points sources located at the nodes of the $Q_x \times Q_y$ regular grid enclosing the conductor (analogously with P-FFT algorithm [10]) [see Fig. 1 and (13)]. The approximate matrix of far interactions $\mathcal{L}^{\text{far}} \simeq \mathbf{L}^{\text{far}}$ with its elements calculated according to (8) can then be stated as the following difference of the matrices of *near* interactions and *all* interactions, respectively, each formed by the interactions of the point-source-based basis and test function representations (8):

$$\mathcal{L}^{\text{far}} = \mathcal{L}^{\text{all}} - \mathcal{L}^{\text{near}}. \tag{9}$$

Substitution of the approximation (9) for the matrix of far interactions into the SLAE (5) produces the following matrix equation amenable to fast iterative solution:

$$(\sigma^{-1}\mathcal{I} + \imath\omega\mathbf{L}^{\text{near}} - \imath\omega\mathcal{L}^{\text{near}} + \imath\omega\mathcal{L}^{\text{all}})\cdot\mathbf{j} = \mathbf{v}^{\text{p.u.l.}}. \tag{10}$$

In SLAE (10), matrix $\hat{\mathcal{L}}^{\text{near}} = \mathbf{L}^{\text{near}} - \mathcal{L}^{\text{near}}$ is the difference between the near interactions' matrix $\mathbf{L}^{\text{near}}$ computed using the original MoM basis and test functions and the near interactions matrix $\mathcal{L}^{\text{near}}$ computed using point-source representations of the basis and test functions (8).

The matrix of all interactions $\mathcal{L}^{\text{all}}$ in (10) can be written as the following product:

$$\mathcal{L}^{\text{all}} = \Lambda^t \cdot \mathbf{G}\cdot\Lambda \tag{11}$$

where 4-D Toeplitz matrix **G** of point-to-point interactions

$$\mathbf{G} = \begin{bmatrix} G_{1111} & \cdots & G_{11Q_xQ_y} \\ \vdots & \ddots & \vdots \\ G_{Q_xQ_y11} & \cdots & G_{Q_xQ_yQ_xQ_y} \end{bmatrix}. \tag{12}$$

Elements $G_{q_xq_yq'_xq'_y}$ of 4-D matrix **G** in (12) are defined as the field produced at observation locations $\boldsymbol{\rho}_{q_xq_y}$ by the point sources located at $\boldsymbol{\rho}'_{q'_xq'_y}$

$$\begin{aligned} G_{q_xq_yq'_xq'_y} &= \mathrm{G}(\boldsymbol{\rho}_{q_xq_y}, \boldsymbol{\rho}_{q'_xq'_y}) \\ &= \mathrm{g}(|\boldsymbol{\rho}_{q_xq_y} - \boldsymbol{\rho}_{q'_xq'_y}|) \\ &= -1/(2\pi)\ln\left[\sqrt{\Delta x^2(q_x - q'_x)^2 + \Delta y^2(q_y - q'_y)^2}\right] \end{aligned} \tag{13}$$

where $q_x, q'_x = 1, \ldots, Q_x$, $q_y, q'_y = 1, \ldots, Q_y$, and Δx, Δy are the steps of the regular grid over x- and y-coordinates. Due to translational invariance of the Green's function G which manifests itself in dependence of G on difference of the position vectors $\boldsymbol{\rho}_{q_xq_y} - \boldsymbol{\rho}_{q'_xq'_y}$ all $Q_x^2\ Q_y^2$, the elements of the 4-D matrix **G** can be produced by only Q_xQ_y elements of 2-D generator matrix **g** as $G_{q_xq_yq'_xq'_y} = g_{q_x-q'_x, q_y-q'_y}$.

The sparse matrix Λ pre- and post-multiplying Toeplitz matrix **G** in (11) has the meaning of converter of N weights of the MoM basis functions **j** to Q_xQ_y coefficients of the point sources on the regular Cartesian grid; hence, it has the following structure with Q_xQ_y rows and N columns:

$$\Lambda = \begin{bmatrix} \Lambda_{11}^1 & \cdots & \Lambda_{11}^N \\ \vdots & \ddots & \vdots \\ \Lambda_{Q_xQ_y}^1 & \cdots & \Lambda_{Q_xQ_y}^N \end{bmatrix}. \tag{14}$$

Each of the N columns in Λ has only Q^2 nonzero elements.

The product $\mathbf{G}\cdot\Lambda\cdot\mathbf{j}$ has the physical meaning of the magnetic vector potential $\widetilde{\boldsymbol{A}}$ at the locations $\boldsymbol{\rho}_{q_xq_y}$ on the grid by the basis functions $\tilde{b}$ projected onto the same grid

$$\widetilde{A}_{q_xq_y} = \widetilde{A}(\bar{\boldsymbol{\rho}}_{q_xq_y}) = \sum_{s=1}^{N} \mathrm{j}_s \sum_{q'_x=1}^{Q_x}\sum_{q'_y=1}^{Q_y} \Lambda_{q'_xq'_y}^{s}\, \mathrm{G}(\boldsymbol{\rho}_{q_xq_y}, \boldsymbol{\rho}'_{q'_xq'_y}) \tag{15}$$

where $q_x = 1, \ldots, Q_x$ and $q_y = 1, \ldots, Q_y$. Summing magnitudes of all point source basis functions on the grid and denoting it as $\mathcal{J}_{q'_xq'_y}$, that is

$$\mathcal{J}_{q'_xq'_y} = \sum_{s'=1}^{N} \mathrm{j}_{s'} \Lambda_{q'_xq'_y}^{s'} \tag{16}$$

we rewrite the scattered field in the form of 2-D convolution

$$\begin{aligned} \widetilde{A}_{q_x,q_y} &= \sum_{q'_x=1}^{Q_x}\sum_{q'_y=1}^{Q_y} G_{q_xq_yq'_xq'_y} \sum_{s'=1}^{N} \mathrm{j}_{s'}\Lambda_{q'_xq'_y}^{s'} \\ &= \sum_{q'_x=1}^{Q_x}\sum_{q'_y=1}^{Q_y} g_{|q_x-q'_x|,|q_y-q'_y|}\mathcal{J}_{q'_xq'_y} \end{aligned} \tag{17}$$

where $G_{q_xq_yq'_xq'_y} = g_{|q_x-q'_y|,|q_x-q'_y|} = \mathrm{G}(\boldsymbol{\rho}_{q_xq_y}, \boldsymbol{\rho}'_{q'_xq'_y})$ is the 4-D Toeplitz matrix and g is its 2-D generating matrix (generator). Introducing single index n to identify each observation point $\boldsymbol{\rho}_{q_xq_y}$ and index m to identify each source point $\boldsymbol{\rho}'_{q'_xq'_y}$ such that $G_{n,m}^{\text{2-D}} = G_{q_xq_yq'_xq'_y}$, we can represent the 4-D Toeplitz matrix $\mathbf{G} = G_{q_xq_yq'_xq'_y}$ as the 2-D matrix $\mathbf{G}^{\text{2-D}} = G_{n,m}$, where $n = 1, \ldots, Q_xQ_y$ and $m = 1, \ldots, Q_xQ_y$.

III. TT Preliminaries and Notations

In the description of the TT-related algebra, we use a lowercase letter (e.g., a) to represent a scalar, a bold lowercase letter (e.g., **a**) to represent a vector, a bold uppercase letter (e.g., **G**) to represent a matrix, and a calligraphic letter (e.g., $\mathcal{G}$) to represent a high-dimensional tensor. The ith element in **g** is a scalar denoted as g_i. The scalar g represents the number of elements in **g**. In the sequel, such scalars will be determining the number of elements in multidimensional matrices (tensors) along particular dimensions.

Given a vector **g**, the binary conversion of all elements in **g** is represented as $\widetilde{\mathbf{g}}$ (e.g., for $\mathbf{g} = [1; 3; 4]$)

$$\widetilde{\mathbf{g}} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{18}$$

and the third binary index of the second element can be extracted as $\widetilde{g}_{2,3}$, which is 0.[1]

A TT representation of d-dimensional tensor $\mathcal{G}(i_1, i_2, i_3, \ldots, i_d)$ has the following format [19], [22]:

$$\begin{aligned} \mathcal{G}(i_1, i_2, \ldots, i_d) &= \sum_{\alpha_1,\ldots,\alpha_{d-1}} \mathcal{G}_1(i_1, \alpha_1)\mathcal{G}_2(\alpha_1, i_2, \alpha_2) \\ &\quad \ldots \mathcal{G}_{d-1}(\alpha_{d-2}, i_{d-1}, \alpha_{d-1})\mathcal{G}_d(\alpha_{d-1}, i_d) \end{aligned} \tag{19}$$

where i_k is the kth level index (dimension) of the tensor, and TT factor index k being $1, 2, \ldots, d$. Each 2-D or 3-D carriage $\mathcal{G}_k$ is known as a TT factor (or TT core), and the auxiliary index α_k determines the kth TT rank. The TT rank r_k is related to the kth unfolding matrix $\mathbf{G}^k$, which is defined as the following:

$$\mathbf{G}^k(\overline{i_1 i_2 \ldots i_k}, \overline{i_{k+1} \ldots i_d}) = \mathcal{G}(i_1, i_2, \ldots, i_d). \tag{20}$$

In the subsequent description, the *TT array* will be denoting the original 4-D Toeplitz matrix $\mathbf{G} = \{G_{q_xq_yq'_xq'_y}\} = \mathcal{G}(i_1, i_2, \ldots, i_d)$ represented at the TT compressed vector format (19) in quad-based indexing of the elements (analogously with four-level quad-tree representation of matrix $\bar{\bar{A}}$ in [22, eq. (8)] and [22, Fig. 2]). On the other hand, the *TT-matrix* term will be used to denote the TT-compressed matrix in the form $\mathcal{G}(n_1m_1, n_2m_2, n_3m_3, \ldots, n_dm_d)$, which contains two modes in each TT factor (denoted as $A_{bin}(n_1, n_2, n_3, \ldots, n_d, m_1, m_2, m_3, \ldots, m_d)$ in [22]). That is, the source and observer binary-tree-based indexes are used to identify the elements in the TT matrix. For example, m_k is the kth bit in the hierarchical binary

[1]Bits from left to right represent low-to-high dimensions.

representation (binary-tree-based) of the original 2-D matrix $\mathbf{G}^{\text{2-D}}$ columns. For brevity, we use $\hat{g}$ to represent a vector consisting of all elements g_k and denote elements $\mathcal{G}(n_1m_1, n_2m_2, n_3m_3, \ldots, n_dm_d)$ as $\mathcal{G}_{\hat{n},\hat{m}}$.

Quantized TT (QTT) decomposition [29] attempts to maximize the number of dimensions in the representation of a given matrix and, as a result, to minimize the size of tensor factors over each of those respective dimensions. Hence, QTT produces such tensor factors (also known as cores) with minimum size over each of their dimensions, which are equal to the minimum possible prime number (conventionally 2, i.e., binary variation (mode) over each dimension). In this article, we use mode size 2 for each bit n_k and m_k in binary indexing of 4-D Toeplitz matrix elements $\mathbf{G}$ over observer and source points as further explained in Section IV.

IV. Representation of 4-D Toeplitz Matrix for Binary Mode QTT Decomposition

Conventionally, the $Q_x \times Q_y$ 2-D generator matrix $\mathbf{g}$ of the 4-D Toeplitz matrix $\mathbf{G}$ is stored in $O(Q_xQ_y)$ memory. It is known as circulant matrix (tensor) in the CG-FFT and P-FFT methods [1], [2], [10], [11]. In the hierarchically partitioned 4-D Toeplitz matrix, QTT explores the low-rank relationship between each level of the hierarchy and stores the least possible number of elements required at each level [20], [22]. Due to the special structure of the Toeplitz matrix, the number of required elements from each level (also known as tensor ranks) is bounded by a low constant (also known as TT rank) in the low-frequency and static regimes and the group electrical size at the given level in the full-wave regime. Therefore, under the assumption of constant TT ranks, QTT can compress the 4-D matrix $\mathbf{G}$ into a d-dimensional TT matrix $\mathcal{G}(n_1m_1, n_2m_2, n_3m_3, \ldots, n_dm_d)$ or $\mathcal{G}_{\hat{n},\hat{m}}$ with time and memory complexities of $O(\log(Q_xQ_y))$, where n_k and m_k are the kth level tensor dimension size, k being $1, 2, \ldots, d$. Note that since there are Q_xQ_y points on the grid, $Q_xQ_y = 2^d$. Due to the intrinsic characteristic of low-rank QTT demanding the same number of elements along each dimension, the computational domain must be square-shaped with an equal number of points introduced over x- and y-coordinates, i.e., $Q_x = Q_y$. The 2-D grid of points is then recursively bisected to represent each point on the 2-D uniform grid as a leaf in the binary tree. Therefore, each grid point can be addressed as a binary number $\hat{n}$, where $\hat{n} = (n_1n_2 \ldots n_d)$ (see [22, Fig. 2] for details).

The $Q_x^2Q_y^2$ elements $G_{q_xq_yq'_xq'_y}$ of the 4-D matrix $\mathbf{G}$ can be represented as $G_{\hat{n},\hat{m}}$. Interaction of each observation point $\hat{n}$ with source point $\hat{m}$ forming an element $G_{\hat{n},\hat{m}}$ can be equivalently indexed as $G_{\text{bin}}(n_1n_2 \ldots n_d, m_1m_2 \ldots m_d)$ according to the grid bisectioning-based indexing. Observation grid point location can be identified by its index n according to the bits of binary ID $\hat{n}$ as $n = n_12^0 + n_22^1 + \cdots + n_d2^{d-1} + 1$. Likewise, source point location can be identified by its index m according to the bits of the binary ID $\hat{m}$ as $m = m_12^0 + m_22^1 + \cdots + m_d2^{d-1} + 1$. Subsequently, $\mathbf{G}_{\text{bin}}$ can be cast into the TT form $\mathcal{G}$ using the SVD-based TT decomposition algorithm [22, Fig. 4]. Such an approach is inefficient as it requires construction and storage of full $\mathbf{G}_{\text{bin}}$ beforehand, followed by successively reshaping $\mathbf{G}_{\text{bin}}$ into unfolding matrix $\mathbf{G}^k_{\text{bin}}(\overline{n_1m_1n_2m_2 \ldots n_km_k}, \overline{n_{k+1}m_{k+1} \ldots n_dm_d})$, storing the multiplication of U_k and S_k as TT core $\mathcal{G}_k$, and updating $\mathbf{G}^{k+1}_{\text{bin}}$ with V'_k, where U_k, S_k, and V_k are the matrix factors generated by applying SVD on the kth unfolding matrix. A more practical method to obtain such tensor representation is by the alternating minimum energy normalization (AMEN) cross algorithm (also known as *TT-cross* algorithm) [20], [22], [27]. The latter makes initial guess for all TT cores and passes those randomly generated elements through all TT cores successively, while using the quasi-maximum volume algorithm (QMVA) [20] to search for the best quality submatrices for all the TT cores. Either method can convert $\mathbf{G}_{\text{bin}}$ into the following d-dimensional TT matrix $\mathcal{G}$ of products of 3-D or 4-D tensors (TT carriages) within predefined tolerance, which is used as the criteria for their low-rank definition:

$$\begin{aligned}\mathcal{G}(n_1m_1, n_2m_2, \ldots, n_dm_d) &= \sum_{\alpha_1,\ldots,\alpha_{d-1}} \mathcal{G}_1(n_1, m_1, \alpha_1) \\ &\quad \times \mathcal{G}_2(\alpha_1, n_2, m_2, \alpha_2) \ldots \mathcal{G}_{d-1}(\alpha_{d-2}, n_{d-1}, m_{d-1}, \alpha_{d-1}) \\ &\quad \times \mathcal{G}_d(\alpha_{d-1}, n_d, m_d). \end{aligned} \tag{21}$$

In (19), each 3-D or 4-D matrix term $\mathcal{G}_k$ is a TT carriage (core), k is the core index, n_k and m_k are the kth modes (dimensions) of the $\mathcal{G}_k$ tensor core, and α_k being $0, 1, \ldots, r_k - 1$ and is the intermediate dimension defined by the kth TT rank.

V. TT-Vector Multiplication Formulation

A. Conventional TT-Vector Multiplication

One way to interpret matrix-vector multiplication (MVM) of $\mathbf{G}$ with the vector of point-source magnitudes $\mathbf{\Lambda} = \mathcal{J}$ is to consider it as 2-D discrete convolution of the signal $\mathbf{\Lambda}$ with the signal $\mathbf{g}$. When the vector of sources (which is a 2-D matrix reshaped into a 1-D vector) convolves with each row of the 4-D Toeplitz matrix (each row is 2-D slice of the 4-D matrix reshaped into a 1-D vector), it results in a 1-D vector of the scattered field (reshaped from 2-D matrix into 1-D), which consists of the sum of elementwise multiplications between the vector of sources and each row of the Toeplitz matrix. This generates the same vector as the conventional MVM. Instead of input vector convolving through each row of the matrix, TT-vector multiplication (TTVM) [29] can be viewed as the process of input vector convolving through d TT factors successively. The TT decomposition compresses the original Toeplitz-structured matrix that contains the row dimension n and the column dimension m by folding its both dimensions into a hierarchical format. Consequently, the ith TT core has four dimensions and contains $r_i \times \mathsf{n}_i \times \mathsf{m}_i \times r_{i+1}$ elements, where r_i and r_{i+1} are the current and next core ranks, and row and column dimensions at the ith level of hierarchy are n_i and m_i (in our case, $\mathsf{n}_i = \mathsf{m}_i = 2$). TT ranks r_1 and r_{d-1} are 1. Therefore, the first and last TT factors are 3-D tensors.

In TTVM, the input vector convolves through all TT factors successively. A TTVM process with $d = 4$ is shown in Fig. 2. The method starts with matrix–matrix multiplication

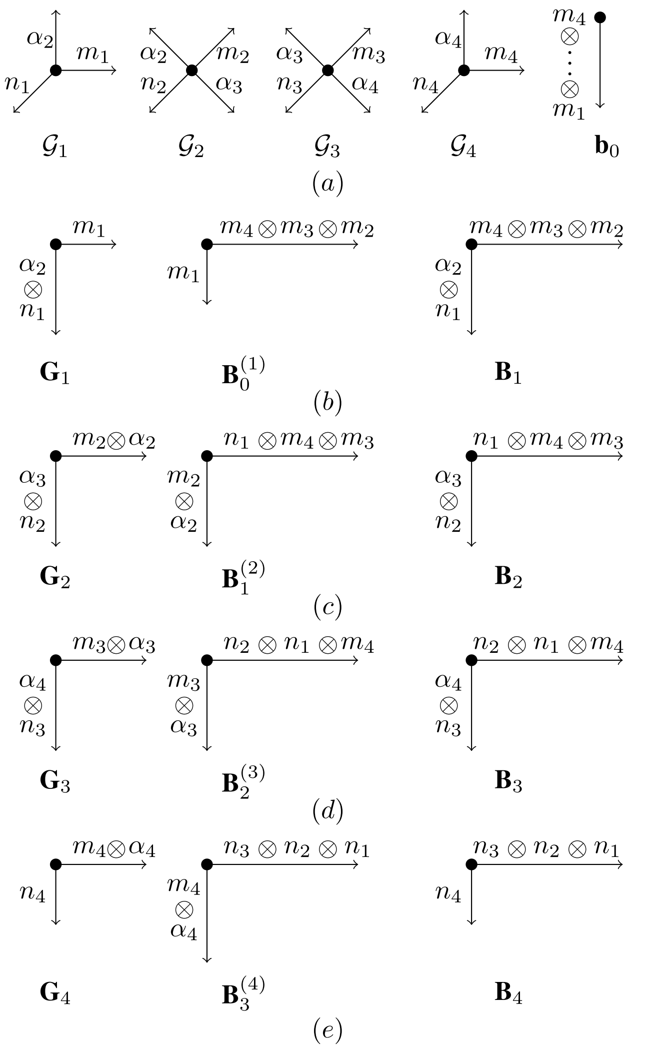


Fig. 2. Multiplication of 4-D TT with vector $\mathbf{b}_0$ in conventional (sparsity-unaware) TTVM. The lengths of TT rank indices α_1 and α_5 are assumed to be 1 (not shown). Each arrow represents an individual mode (dimension) of a tensor, and 1-arrow and 2-arrow cells represent vector and matrix, respectively. 3-arrow and 4-arrow cells represent 3-D and 4-D tensors, respectively. (a) Original 4-D TT and input vector $\mathbf{b}_0$ $[m_4 \otimes m_3 \otimes m_2 \otimes m_1, 1]$. Steps (b)–(d) illustrate multiplications of the TT cores with appropriately reshaped matrices according to Algorithm 1. (e) Procedure that produces output $\mathbf{B}_4$ $[n_4 \otimes n_3 \otimes n_2 \otimes n_1, 1]$ after four multiplications with all TT factors successively according to Algorithm 1. The sought output vector $\mathbf{b}_5 = \mathcal{G} \cdot \mathbf{b}_0$ is obtained by directly reshaping $\mathbf{B}_4$.

of reshaped first TT core $\mathcal{G}_1$ and matrix format of input vector $\mathbf{b}_0$, which results in a 2-D matrix $\mathbf{B}_1$ of dimensions $r_2\mathsf{n}_1 \times \mathsf{m}_4\mathsf{m}_3\mathsf{m}_2$ with the exclusion of mutual dimension m_1 [see Fig. 2(b)]. Therefore, the intrinsic concept of TTVM is to perform d inner products each over dimensions m_i and r_i, $i = 1, \ldots, d$. In Fig. 2(d), $\mathbf{B}_4$ contains only TT matrix row indices n_i, where $i = 1, 2, 3, 4$. Reshape on $\mathbf{B}_4$ casts it into vector format $\mathbf{b}_5$, which is the same as the result of conventional MVM, provided TT cores exactly represent the original matrix.

Algorithm 1 [21] presents a complete implementation for TTVM focusing on the illustration of dimensional indices' variation throughout all d multiplications. For clarity, all TT factors are reshaped into matrix format (line 6 in Algorithm 1). This can be alternatively done with maintained

Algorithm 1 TTVM per Fig. 2

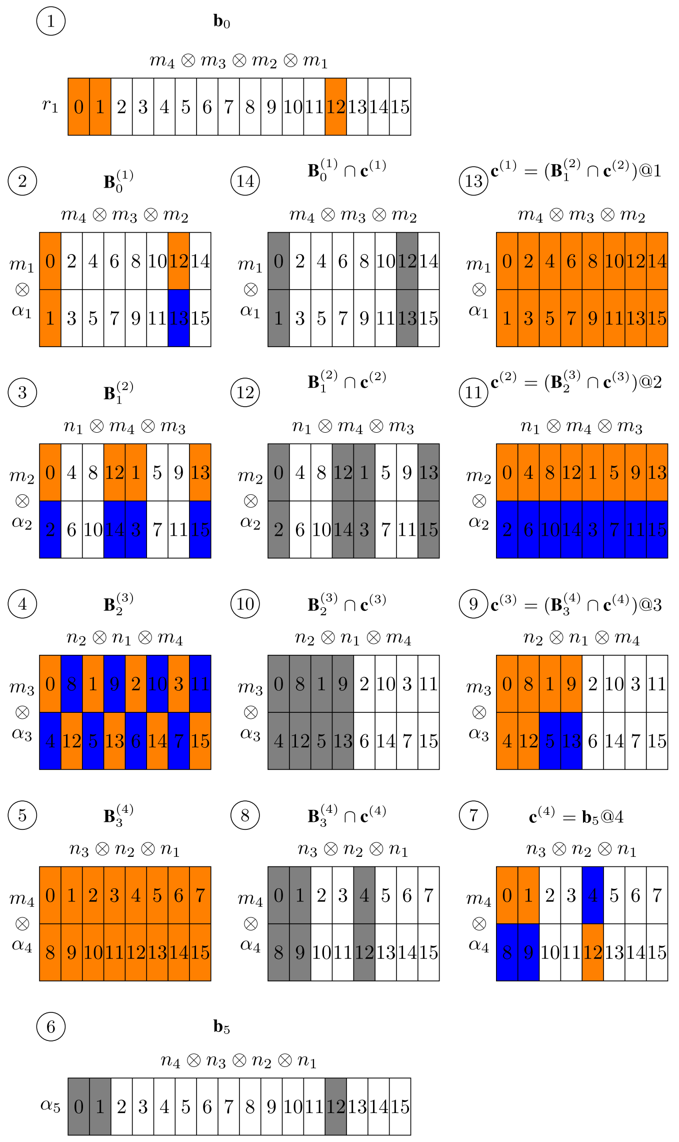
```
1: Inputs: TT factors $\mathcal{G}$ (as in (19)), and input vector $\mathbf{b}_0$.
2: Outputs: A vector $\mathbf{b}_{d+1}$ as the result of TT compressed
   matrix multiplied with the input vector, $\mathbf{b}_{d+1} = \mathcal{G} \cdot \mathbf{b}_0$.
3: Reshape vector $\mathbf{b}_0$ into $d$ dimensional array of correspond-
   ing mode side at each dimension, and save it as $\mathbf{B}_0$.
4: Reshape $\mathbf{B}_0$ into $\mathbf{B}_0^{(1)}$ $[\mathsf{m}_1, \mathsf{m}_d \ldots \mathsf{m}_2]$.
5: for $k = 1 : d$ do
6:     Reshape $\mathcal{G}_k$ into $\mathbf{G}_k$ $[r_{k+1}\mathsf{n}_k, \mathsf{m}_k r_k]$.
7:     Update $\mathbf{B}_k = \mathbf{G}_k \cdot \mathbf{B}_{k-1}^{(k)}$.
8:     Reshape $\mathbf{B}_k$ to $\mathbf{B}_k^{(k+1)}$ $[\mathsf{n}_k, \mathsf{n}_{k-1} \ldots \mathsf{n}_1\mathsf{m}_d \ldots \mathsf{m}_{k+1}r_{k+1}]$.
9:     Transpose $\mathbf{B}_k^{(k+1)}$ to $\mathbf{B}_k^{(k+1)}$
       $[\mathsf{n}_{k-1} \ldots \mathsf{n}_1\mathsf{m}_d \ldots \mathsf{m}_{k+1}r_{k+1}, \mathsf{n}_k]$.
10:    Reshape $\mathbf{B}_k^{(k+1)}$ to $\mathbf{B}_k^{(k+1)}$
       $[\mathsf{m}_{k+1}r_{k+1}, \mathsf{n}_k\mathsf{n}_{k-1} \ldots \mathsf{n}_1\mathsf{m}_d \ldots \mathsf{m}_{k+2}]$.
11: end for
12: Reshape $\mathbf{B}_d$ into $\mathbf{b}_{d+1}$.
13: return
```

high-dimensional tensor. Reshapes and transpositions are necessary in order to adjust the dimensions of the successively obtained matrices and perform their multiplications.

B. Sparsity-Aware TTVM

MVM is required by the iterative matrix solvers, such as CG and GMRES. In the P-TT algorithms, the currents in the object of interest are projected on the regular grid. In the vicinity of each element, the grid points assume nonzero magnitudes of the point sources. TT stores Toeplitz-structured matrix corresponding to the interactions between point sources located at the nodes of the regular grid in $O(\log(N))$ memory at low frequencies and in $O(r_{\max}^2 \log N)$ memory at high frequencies, $r_{\max} \approx \sqrt{N}$ when N is the number of grid points in 2-D. The important property of the TT is that it allows access to a particular portion of compressed data without recovering the full matrix from it. Furthermore, the way TT accesses data can be exploited to account for the sparsity of the grid population by the sources (nonzero magnitude point sources in the proposed P-TT method) when TTVM is evaluated. Here, we show how a portion of compressed data pertinent to evaluation of the TTVM can be effectively accessed in the sparsity-aware manner in order to reduce the computational cost.

We choose QTT as a compression scheme of the 4-D Toeplitz matrix $\mathbf{G}$ formatted according to the binary addressing of its elements, which describes point-to-point interactions on the regular grid. In Fig. 2, the behavior of dimensional indices is illustrated through each multiplication stage of TTVM. We will show in the following that the element indices from matrix factors $\mathbf{B}_k$ at every successive kth multiplication step can be predicted beforehand in accord with the sparsity pattern. If the input vector $\mathbf{b}_0$ (see Fig. 2) includes empty (zero) elements due to the sparsity in grid population instead being a full vector, it is important to exclude this nonessential information from it in order to speed up the



1
$\mathbf{b}_0$
$m_4 \otimes m_3 \otimes m_2 \otimes m_1$
r_1
2
$\mathbf{B}_0^{(1)}$
14
$\mathbf{B}_0^{(1)} \cap \mathbf{c}^{(1)}$
13
$\mathbf{c}^{(1)} = (\mathbf{B}_1^{(2)} \cap \mathbf{c}^{(2)})@1$
$m_4 \otimes m_3 \otimes m_2$
$m_1 \otimes \alpha_1$
3
$\mathbf{B}_1^{(2)}$
12
$\mathbf{B}_1^{(2)} \cap \mathbf{c}^{(2)}$
11
$\mathbf{c}^{(2)} = (\mathbf{B}_2^{(3)} \cap \mathbf{c}^{(3)})@2$
$n_1 \otimes m_4 \otimes m_3$
$m_2 \otimes \alpha_2$
4
$\mathbf{B}_2^{(3)}$
10
$\mathbf{B}_2^{(3)} \cap \mathbf{c}^{(3)}$
9
$\mathbf{c}^{(3)} = (\mathbf{B}_3^{(4)} \cap \mathbf{c}^{(4)})@3$
$n_2 \otimes n_1 \otimes m_4$
$m_3 \otimes \alpha_3$
5
$\mathbf{B}_3^{(4)}$
8
$\mathbf{B}_3^{(4)} \cap \mathbf{c}^{(4)}$
7
$\mathbf{c}^{(4)} = \mathbf{b}_5@4$
$n_3 \otimes n_2 \otimes n_1$
$m_4 \otimes \alpha_4$
6
$\mathbf{b}_5$
$n_4 \otimes n_3 \otimes n_2 \otimes n_1$
α_5

Fig. 3. Forward filtering (steps ①–⑤) and backward filtering (steps ⑥–⑭) of the matrices in four-level TTVM (Fig. 2) according to input and output vector sparsities. All TT ranks are assumed as 1 for simplicity.

evaluation of the TTVM. If the empty elements are not eliminated, unnecessary operations of multiplications by zeros persist throughout the successive multiplication process. Also, the output vector elements are of interest only at the grid points where the scattered field needs to be evaluated to compute its reaction with the discrete representation of the test functions, i.e., in the vicinity of the object. In other words, the sparsity pattern in the output vector is the same as that of the input vector. This sparsity pattern in the output vector can also be accounted for in the construction of the TT cores to eliminate unnecessary operations and memory use. Therefore, the optimal indices' selection consists of two subprocesses, namely forward filtering and backward filtering, for the processes starting from the input and output vectors, respectively.

Fig. 3 provides a detailed illustration of how data permutations are performed according to the sparsity pattern. Such permutations eliminate unnecessary operations from the TTVM. Due to the fact that an iterative solver requires repetitive evaluation of the MVMs (or TTVMs in the P-TT algorithm), it is necessary to preprocess sparsity-related information prior to the initiation of the iterative process rather than repeatedly reevaluating it during the iterations. This preprocessing is also important for the purpose of memory saving.

Given all nonzero entries in the input vector $\mathbf{b}_0$ (see Fig. 3 and corresponding steps of the TTVM in Fig. 2 and Algorithm 1), the sparsity pattern in matrix $\mathbf{B}_0^{(1)}$ in Fig. 2 can be predicted by arranging given nonzero values according to the reshape transforming vector $\mathbf{b}_0$ into matrix $\mathbf{B}_0^{(1)}$. However, noncontributing zero elements must be introduced when the matrix columns have missing elements (gaps) that are not provided by the preceding array (for example, blue cell "13" is introduced into matrix $\mathbf{B}_0^{(1)}$ in Fig. 3). These zero values are padded into the matrix structure to ensure completeness of its columns, since the matrix is stored in the column-by-column manner. Matrix $\mathbf{B}_0^{(1)}$ is subsequently multiplied with the first core $\mathbf{G}_1$ of TT decomposed matrix $\mathbf{G}$ (see Fig. 2). After this multiplication, the zero-padded cell [e.g., cell "13" in matrix $\mathbf{B}_0^{(1)}$] will acquire nonzero values from their adjacent column neighbor values [e.g., cell "12" in matrix $\mathbf{B}_0^{(1)}$]. Noncontributing (zero) elements are consistently introduced both due to multiplication with the corresponding core as well as the reshaping [see blue elements in matrix $\mathbf{B}_1^{(2)}$]. In the example of Fig. 3, this leads to the matrix $\mathbf{B}_2^{(3)}$ to include the elements from all the grid points. Multiplication of matrix $\mathbf{B}_2^{(3)}$ with the matrix $\mathbf{G}_3$ of the reshaped core $\mathcal{G}_3$ (see Fig. 3) results in matrix $\mathbf{B}_3^{(4)}$, which has no zero elements. Subsequently, matrix $\mathbf{B}_3^{(4)}$ is multiplied with last reshaped core $\mathbf{G}_4$ to produce vector $\mathbf{b}_5$ of scattered field values on the entire grid. Notice, however, in the output vector $\mathbf{b}_5$, which represents scattered field, 13 values (white cells in Fig. 3) can be removed in accord with its sparsity pattern corresponding to the needed values of the scattered field on the grid. The fact that we can remove the unnecessary 13 values from the output vector $\mathbf{b}_5$ allows us to perform another filtering based on the output vector (scattered field) sparsity. Indeed, the first filtering (forward filtering) was done using only the information about input vector sparsity (i.e., sparsity of the sources on the grid), which lacks the information about the sparsity of the output vector (i.e., sparsity of the required scattered field on the grid). In order to propagate this output vector sparsity in the TTVM process, "backward filtering" is introduced.

The idea of this "backward filtering" is to perform the above-described "forward filtering," with the awareness of which output vector values will be required rather than computing all of them (to avoid unnecessary computations of the scattered field on the entire grid). Since there exists a significant number of elements in matrix factors $\mathbf{B}_k^{(k+1)}$ that after multiplying with the TT factors do not contribute to the elements of interest in the final output vector $\mathbf{b}_5$, we first reshape sparsity mask of the output vector $\mathbf{b}_5$ into the format of matrix $\mathbf{B}_3^{(4)}$ to produce the mask denoted as $\mathbf{c}^{(4)} = \mathbf{b}_5@4$ in Fig. 3. Orange elements in $\mathbf{c}^{(4)}$ indicate nonzero elements coming directly from pattern $\mathbf{b}_5$ while blue elements denote elements forced into $\mathbf{c}^{(4)}$ pattern by the need to complete the columns.

Next, we find the intersection of the nonzero elements' pattern in matrix $\mathbf{B}_3^{(4)}$ and output vector sparsity pattern $\mathbf{c}^{(4)}$. This intersection of the sparsity patterns is labeled as $\mathbf{B}_3^{(4)} \cap \mathbf{c}^{(4)}$ in Fig. 3. Note that since the matrix $\mathbf{B}_3^{(4)}$ is full, the sparsity in intersection $\mathbf{B}_3^{(4)} \cap \mathbf{c}^{(4)}$ is dictated by the sparsity of $\mathbf{c}^{(4)}$.

The combined sparsity pattern $\mathbf{B}_3^{(4)} \cap \mathbf{c}^{(4)}$ produced at level 4 by the input and output vector sparsities is then reshaped to level 3 (denoted as pattern $\mathbf{c}^{(3)} = (\mathbf{B}_3^{(4)} \cap \mathbf{c}^{(4)})@3$ in Fig. 3). Once again, the orange elements in this reshaped sparsity pattern at level 3 signify the nonzero elements coming directly from $\mathbf{B}_3^{(4)} \cap \mathbf{c}^{(4)}$, while the blue elements result from completion of the columns (padding them with zeros). This pattern is then intersected with the sparsity pattern of matrix $\mathbf{B}_2^{(3)}$, yielding the resultant pattern at level 3 denoted as $\mathbf{B}_2^{(3)} \cap \mathbf{c}^{(3)}$.

The process of reshaping resultant sparsity pattern to the lower level and its intersection with the sparsity pattern induced by the sources in matrices $\mathbf{B}_k^{(k+1)}$ continues to levels 2 and 1 producing sought sparsity patterns $\mathbf{B}_1^{(2)} \cap \mathbf{c}^{(2)}$ and $\mathbf{B}_0^{(1)} \cap \mathbf{c}^{(1)}$ at these levels, respectively (see Fig. 3).

After the resultant sparsity patterns $\mathbf{B}_k^{(k+1)} \cap \mathbf{c}^{(k)}$, $k = 1, \ldots, d$ are found according to the above-described forward and backward filtering steps, they are stored and reused at each TTVM for the construction of matrices $\mathbf{B}_k^{(k+1)}$ in order to avoid operations involving sparsity-induced zero.

VI. Numerical Results

All numerical experiments with up to 200 000 MoM unknowns are conducted on a 2.2-GHz Intel Core i7 processor with 8-GB RAM machine. Larger scale simulations are conducted on a server with 384 Gb of RAM. All CG-TT [22] and P-FFT [10] experiments were single-threaded simulations. It is important to mention that the TTVM can be processed in parallel by splitting the multiplication between each mode of the TT core with the vector into separate processors. The results from different nodes can then be assembled from the nodes responsible for the computation of a part of the resultant vector. Parallelization is out of the scope of this article, however. This simple modification is only mentioned to indicate a simple approach to further improvement of performance reported in this article.

We examine naïve and sparsity-aware P-TT algorithms by comparing them against the P-FFT [10] algorithm. For the MVM of the Toeplitz matrix with a given vector, the P-FFT [10] performs elementwise multiplication between the FFT of the 2-D generator $\mathbf{g}$ and the FFT of the source vector $\mathbf{\Lambda}$, the latter being zero padded to reach the full period of the discrete Fourier transform (DFT) dictated by $\mathbf{g}$. As the problem size grows, computation and storage of the generator $\mathbf{g}$, source matrix $\mathbf{\Lambda}$, and their FFTs can become prohibitively expensive. The $O(\log(N))$ complexity of TT (at low and moderate frequencies) in both forming and storing the Toeplitz matrix and its $O(r_{\max} N)$ and $O((r_{\max})^2 N \log N)$ for TTVM complexities allow to dramatically reduce resources required for solution of a given large-scale problem.

In all the P-TT numerical experiments, we take two layers of the near neighbor elements that are considered to be in the region of near-field interactions with a given observation triangle. The first layer of neighbors for a given observation triangle is formed by all the triangles sharing at least one vertex with it. The second layer of near neighbors is formed by all the triangles sharing at least one vertex with the triangles of the first layer. Indeed, this connectivity-based definition of the near neighbors is proper only in the case of simple convex objects considered in this article. More sophisticated quad-tree-based near neighbor selection methods are required in the general cases.

In all full-wave experiments for the near-field integrals, a six-point third-order quadrature rule is used after the ln-singularity is extracted from it and integrated analytically. The tolerances in constructing TT and termination of the GMRES [16] iterations are fixed as 10^{-6} and 10^{-8}, respectively, unless specified otherwise.

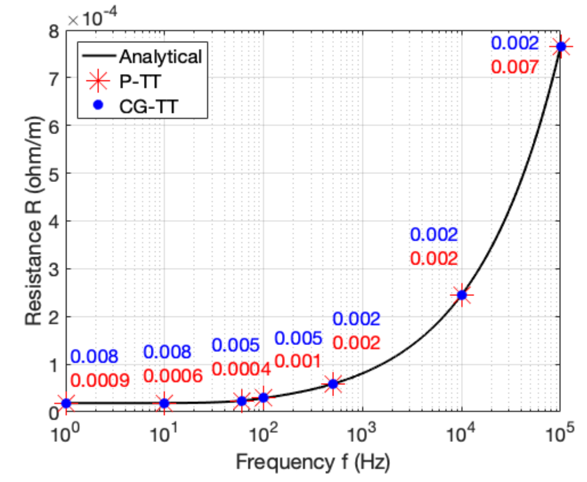
In Sections VI-A–VI-C, we examine the performance of the P-TT and its sparsity-aware formulation through a series of experiments on current flow problems in the cases of single and multiple conductors (Sections VI-A and VI-B) as well as full-wave scattering problems (Section VI-C).

A. QTT Scheme for Current Flow in Single Conductor

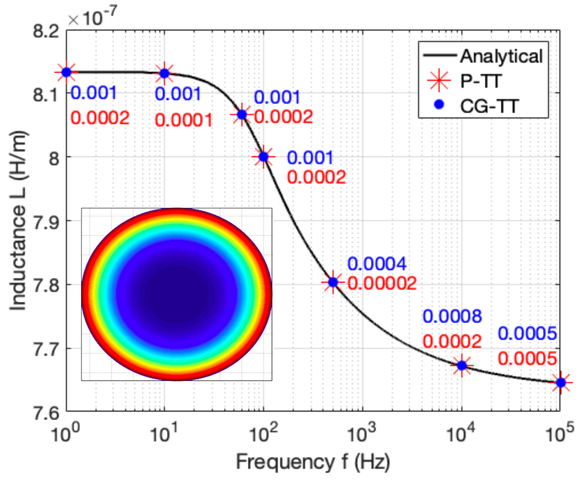
To investigate the performance of the proposed P-TT iterative solver, we consider the current flow problem in a single conductor of the circular cross section with a radius of 0.022 m and the conductivity of 3.57×10^7 S/m. To examine the accuracy of the P-TT algorithm, in Fig. 4(a) and (b), we demonstrate the results of seven experiments for frequencies 1 Hz, 10 Hz, 60 Hz, 100 Hz, 500 Hz, 10 kHz, and 100 kHz and compare them against the analytic solution. Both unstructured mesh of MoM and the structured grid are adjusted for each frequency point to account for the skin depth. To prove the accuracy of using triangle-based mesh over the voxel-based mesh, both CG-TT [22] and P-TT are used for all seven experiments. The voxel-based mesh cannot accurately describe the curvature of circular conductors, when the number of voxels is small. Hence, in low-frequency regimes, if the mesh element size is governed by the skin depth, the P-TT shows higher accuracy than CG-TT [22]. As the number of both triangle and voxel elements' increases, the relative error in both methods becomes comparable. In high-frequency regimes, P-TT shows accurate performance resulting in the relative error of 0.002 and 0.007 compared to the analytic solution at frequencies of 10 and 100 kHz, respectively. In Fig. 4(b), the 60-Hz current distribution of circular conductor discretized into 260 096 triangles is shown.

We use GMRES [16] as an iterative solving scheme for the proposed P-TT. As shown in Fig. 5, at low-frequency regime, P-TT converges at eight iterations and consumes 139.22 s of the total online time (time of GMRES iterations). At a frequency of 100 kHz, convergence is reached after 89 iterations with diagonal preconditioner and takes 1692.97 s in total.

In order to perform a systematic analysis of P-TT performance, we consider a regular grid with



(a)



(b)

Fig. 4. Results of P-TT current flow simulations performed on a circular cross section conductor with a conductivity of $3.57 \cdot 10^7$ S/m. The conductor has a radius of 0.022 m and is centered at the origin. (a) and (b) Relative errors compared against analytic solution for extracted resistances and inductances at the frequencies of 1 Hz, 10 Hz, 60 Hz, 100 Hz, 500, 10 kHz, and 100 kHz are shown for both CG-TT [22] (blue) and P-TT [25] (red). Current flow distribution at 60 Hz obtained by P-TT [25] is shown.

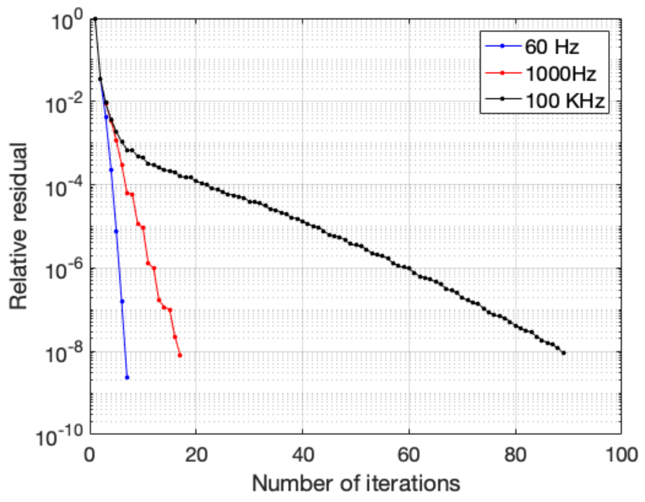


Fig. 5. Convergence rate pattern of P-TT at frequencies of 60 Hz, 1000 Hz, and 100 kHz.

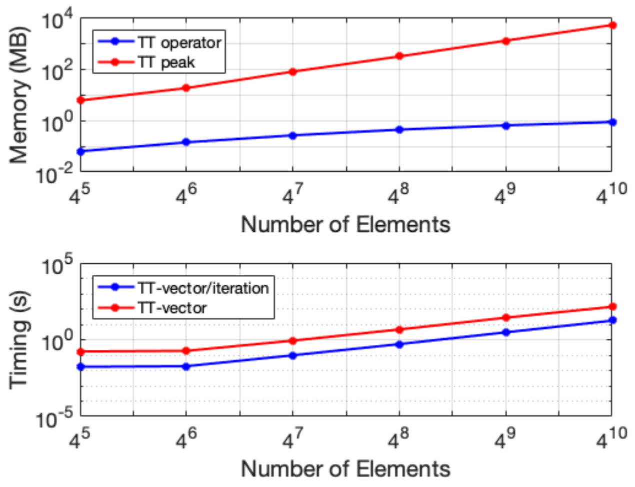


Fig. 6. Top: memory consumptions of both storing TT operator (blue curve) and peak memory usage during TTVM (red curve) for the number of grid elements 4^5, 4^6, 4^7, 4^8, 4^9, and 4^{10}. Bottom: timing consumptions of both TTVM time per iteration (blue curve) and total time (red curve) for the same discretized number of grid elements.

4^5, 4^6, 4^7, 4^8, 4^9, and 4^{10} points and use an approximate ratio of 4 to 1 in relation to the number of triangles[2] taken in the case of 60-Hz excitation. We show that the proposed P-TT has the memory complexity of $O(r_{\max}N)$ and the timing complexity of $O(r^2_{\max}N\log(N))$. Due to the fact that frequency variation only affects the number of iterations to achieve convergence, different frequencies result in the same demonstrations of both peak memory consumption and timing consumption per iteration. Therefore, we only choose the representative 60-Hz results for the complexity study. As shown in Fig. 6 (top), the memory usage of storing TT operator (blue curve) increases linearly, as the number of grid elements increases exponentially. Peak memory consumption during P-TT solving process occurs when the TT core containing the maximum TT rank is being multiplied with the successively obtained vector. Due to the TT tolerance being fixed throughout all experiments, one can observe that the peak memory usage (red curve) increases in accord with the exponentially increased number of grid elements. The timing growth can be observed in Fig. 6 (bottom). When the number of elements grows exponentially, both TT-vector online (red curve) and per iteration time (blue curve) follow the same trend as the elements grow. This is due to the fact that TTVM has the complexity of $O(r^2_{\max}N\log(N))$, and the maximum TT rank is maintained to be the same for all experiments as the tolerance of constructing TT operators is fixed as 10^{-6}.

[2]The ratio between the number of grid points and the number of triangles is approximately 4.

B. Sparsity-Aware P-TT Algorithm in Analysis of MTL

To demonstrate the impact of sparsity in grid population on P-TT performance, we use three-cable MTL with the

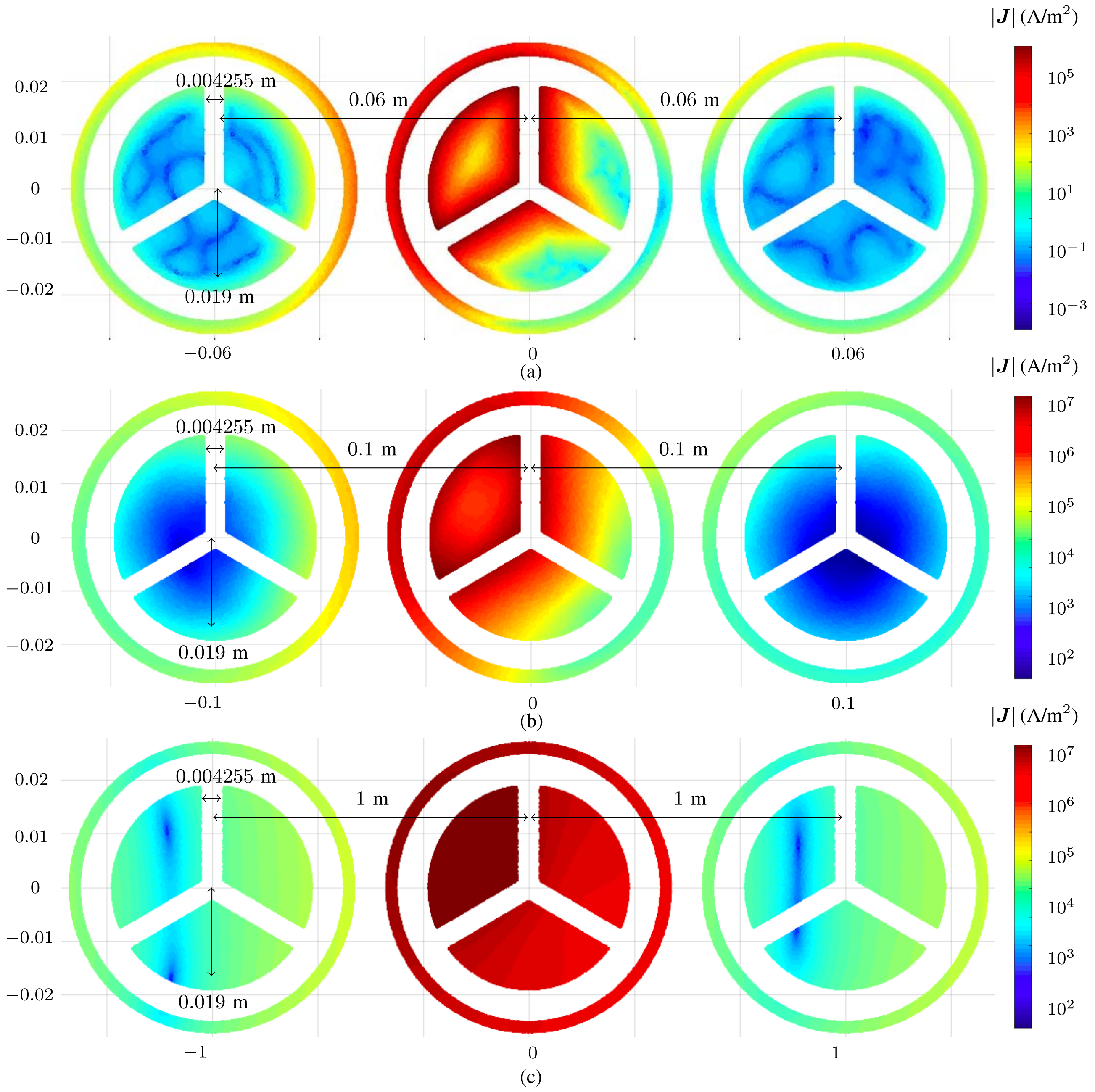
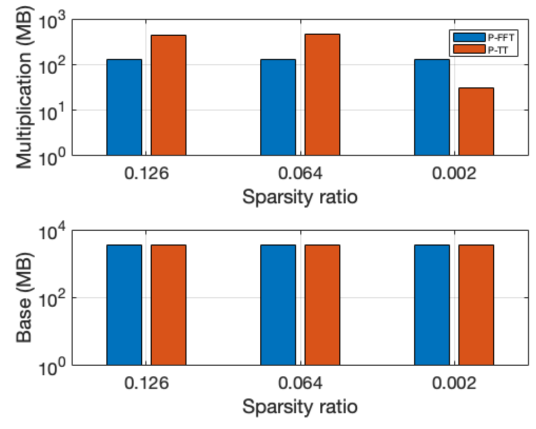


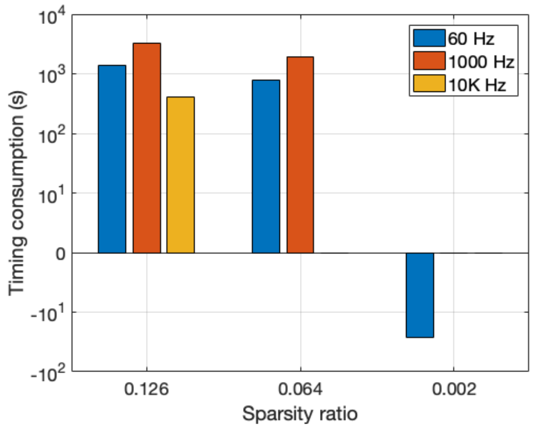
Fig. 7. (a) Current at 10-kHz study and the separation distance of 0.06 m between the centroids of each of the two adjacent conductors. (b) Current at 1-kHz study and the separation distance of 0.1 m between the centroids of each of the two adjacent conductors. (c) Current at 60-Hz study and the separation distance of 1 m between the centroids of each of the two adjacent conductors. All plots are in logarithmic scale.

separation distance between the centroids of adjacent two cables being 0.06, 0.1, and 1 m. Each bundled cable has three sectors with outer and inner radii of the sheath being 0.027 and 0.019 m (see Fig. 7). Conductivity of all conductors is taken to be $3.57 \cdot 10^7$ S/m. A regular grid consisted of 1 048 576 points is introduced over the computational domain. For closely located cables (6-cm separation), the corresponding sparsity ratio[3] is 0.126. The 1 048 576-point grid supports mesh discretizations, which allow for high-frequency simulations including 10 kHz. As a result, for the 3-cable MTL with 0.06-m separation between the cables, we consider three frequencies: 60 Hz, 1000 Hz, and 10 kHz. With the increase in the separation between the cables, the sparsity of the grid population increases. Under the condition of the fixed 1 048 576-point grid, the minimum size of the triangle mesh elements, however, increases with the increased separation between the cables. This limits the maximum frequency of simulations, as the minimum of two triangles is required to sample one skin depth. As a result, only 60- and 1000-Hz frequencies are considered in the 0.1-m separation case (sparsity ratio being 0.064) and only 60-Hz frequency is considered in the 1-m separation case (sparsity ratio being 0.002). Fig. 7(a)–(c) show the current flow distributions computed

[3]Sparsity ratio is defined as the ratio between the number of nonempty grid point and the total number of grid points.



(a)



(b)

Fig. 8. (a) Peak memory consumption in sparsity-aware P-TT and P-FFT [10] at 60 Hz over the sparsity ratio of 0.126, 0.064, and 0.002. The peak memory consumptions are divided into two parts. Top: memory usages of P-FFT [10] and P-TT performing only elementwise multiplication and TTVM, respectively. Bottom: total memory usages of the right-hand side vector multiplying with $\mathbf{L}^{\text{near}}$, $\hat{\boldsymbol{\mathcal{L}}}^{\text{near}}$, and storing Λ, which have the same magnitude for the same frequency, denoted as base memory consumptions. The bars in (b) represent the result of subtraction between P-FFT [10] time consumption and P-TT time consumption. Positive values mean that P-FFT [10] time consumption is less than P-TT time consumption. Sparsity ratio of 0.126 contains 60-Hz, 1000-Hz, and 10-kHz studies, sparsity ratio of 0.064 contains 60-Hz and 1000-Hz studies, and sparsity ratio of 0.002 only contains 60-Hz study. All values in (a) and (b) are plotted in logarithmic scale.

using sparsity-aware P-TT algorithm at 10 kHz, 1000 Hz, and 60 Hz for 0.06-, 0.1-, and 1-m separations, respectively. The extracted admittance matrices are compared against the direct VIE solution [32], [33]. The maximum relative error in the case of 6- and 10-cm separations does not exceed 1% and remains under 5% in the 1-m separation case.

In the three-cable experiments, we investigated various sparsity scenarios with the goal of identifying when the proposed sparsity-aware P-TT algorithm outperforms P-FFT [10]. The three selected meshes in the cases of 6-, 10-, and 1-m separation distances result in grid sparsities of 0.126, 0.064, and 0.002. The memory usage for the two methods is shown in Fig. 8(a) for the frequency of 60 Hz. The peak memory usage is divided into two parts, Fig. 8(a) (top) shows the memory consumptions during MVMs that involves only P-FFT [10] matrices and P-TT matrices (TT cores). Fig. 8(a) (bottom) shows the total memory usage during the MVM with $\mathbf{L}^{\text{near}}$, $\hat{\boldsymbol{\mathcal{L}}}^{\text{near}}$, and Λ, which all remain the same in both P-FFT and P-TT methods under the condition that three layers of near-neighbor elements are taken for each observation triangle. The memory usage in Fig. 8(a) (bottom) is the same for both methods. Therefore, we consider the memory usage by the P-FFT and P-TT algorithms according to the values shown in Fig. 8(a) (top). The memory usage of P-FFT [10] is fixed as 128 MB and P-FFT [10] outperforms P-TT at the sparsity ratios of 12% and 6%. In these cases, the P-TT consumes 400 MB. As sparsity ratio becomes smaller (less than 1%), sparsity-aware P-TT uses less memory (30 MB in the case of 0.002 sparsity ratio) due to elimination of unnecessary information from matrices in the TTVM process. The same trend is shown in Fig. 8(b) for time consumption. In Fig. 8(b), each bar represents the result of subtraction between P-FFT [10] and P-TT time consumptions. A positive value of the difference indicates that P-FFT [10] outperforms P-TT, while the negative values indicate that P-TT outperforms P-FFT. At the sparsity ratios of 0.126 and 0.064, P-FFT [10] outperforms P-TT for all considered frequencies. At the sparsity ratio of 0.126, for example, P-FFT [10] only requires 79 and 445 s in 60- and 1000-Hz cases, while P-TT consumes 197 and 1482 s, respectively, at the same frequencies. When the sparsity ratio becomes less than 1% [the third bar in Fig. 8(b)], P-TT uses less time compared with P-FFT [10] without introducing any additional error other than prescribed tolerance during the TT construction. At the sparsity of 0.002, sparsity-aware P-TT converges under 40 s, while P-FFT [10] consumes 77 s.

The observed impact of grid population sparsity on the performance of the proposed sparsity-aware P-TT indicates that the proposed novel algorithm can be advantageous for simulations that include multiple well-separated conductors in 2-D MTLs, solution of the surface integral equations exhibiting high sparsity in grid population, 3-D magnetostatic interconnect analysis problems [23], and various other practically interesting cases. Meanwhile, as the sparsity ratio becomes smaller, the number of projected elements decreases, which shows the limitation of using FFT-based algorithms.

C. Sparsity-Aware P-TT Algorithm in Solution of Scattering Problems

Performance of the proposed P-TT algorithm is analyzed in the case of full-wave scattering on a cylinder of circular cross section having relative permittivity $\epsilon_r = 4$ and radius of 0.45 m, and situated in free space. The cylinder is centered at the origin. The scatterer is projected onto a square grid with a side length of 1 m. The incident plane wave of 1-V/m magnitude is impinging from the 0° direction. The scatterer is discretized with 100 498 triangles and the MoM basis functions are projected onto the 1 048 576-point Cartesian grid. We conducted three experiments with the frequencies of 0.3, 1, and 2.5. To maintain the same level of accuracy for the increased frequency, we use two, three,

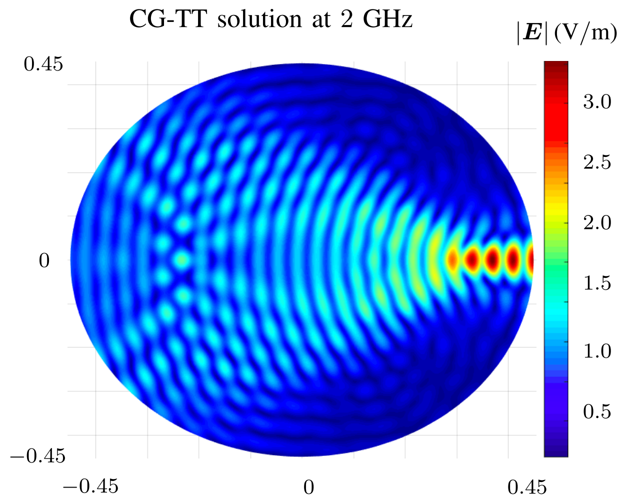
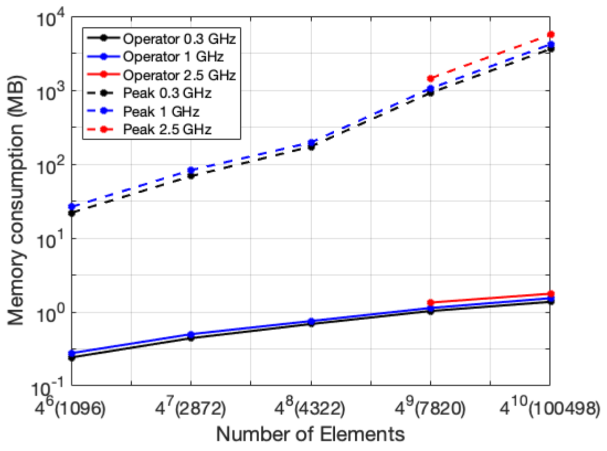


Fig. 9. Magnitude of electric field in circular cylinder of relative permittivity $\epsilon_r = 4$ computed with CG-TT at 2 GHz due to plane wave incident at 0°.

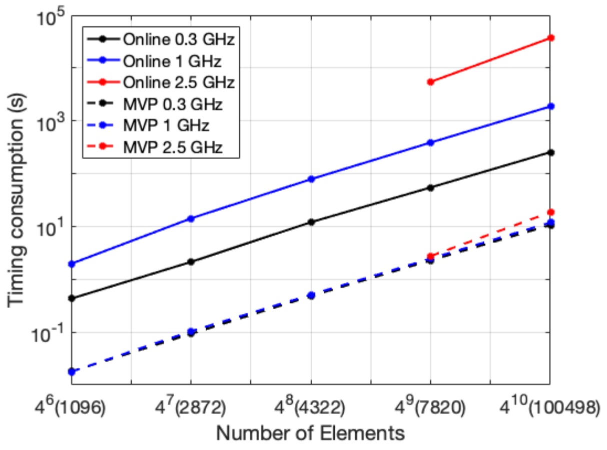
and five layers of connected elements in near interactions for 0.3, 1, and 2.5 GHz experiments, respectively. In low-frequency regime at 0.3 GHz, P-TT converges at 24 iterations after 255.19 s. In high-frequency regime at 2.5 GHz, however, 1988 iterations are required to convergence to prescribed tolerance with the diagonal preconditioner, which takes 36 953.57 s.[4] Fields computed with the P-TT algorithm at frequencies of 0.3, 1, and 2.5 GHz are compared against the Mie series [31]. The average absolute error of $5.74 \cdot 10^{-6}$, $5.83 \cdot 10^{-5}$, and 0.07 V/m was reached at these frequencies, respectively. The field distribution at 2 GHz for a circular cylinder is shown in Fig. 9.

To study the complexity growth of P-TT in case of full-wave scattering problems, the regular grids with 4^6, 4^7, 4^8, 4^9, and 4^{10} points and corresponding triangle meshes with 1096, 2872, 4322, 7820, and 100 498 elements were considered. All the grid cases were considered at 0.3 and 1 GHz, and only 4^9 and 4^{10} grid cases were considered at 2.5 GHz. The performance of P-TT was observed to significantly depend on the number of grid elements rather than the number of triangles. Therefore, we exponentially increased the number of grid points to analyze the CPU time and memory complexity. As shown in Fig. 10(a), as the number of grid elements increases exponentially, the memory used to store TT (solid line) largely increases as $O(\log(N))$ with slight deviation from it at the high frequencies due to the increase in TT ranks. According to the TTVM process described in Section V-A, the peak memory use is observed at the stage of multiplication between the TT core having highest TT rank and the successively obtained vector. With $O(r_{\max} N)$ complexity, the peak memory use grows exponentially with the exponential increase in the number of grid points. A slight deviation is observed again due to the increased TT ranks at high frequencies. In Fig. 10(b), the total online timing (solid line) shows exponential growth, as the number of grid elements increases exponentially. However, the deviation due to increase

[4]The number of iterations can be substantially reduced by using more effective preconditioners, which will be addressed in the future work.



(a)

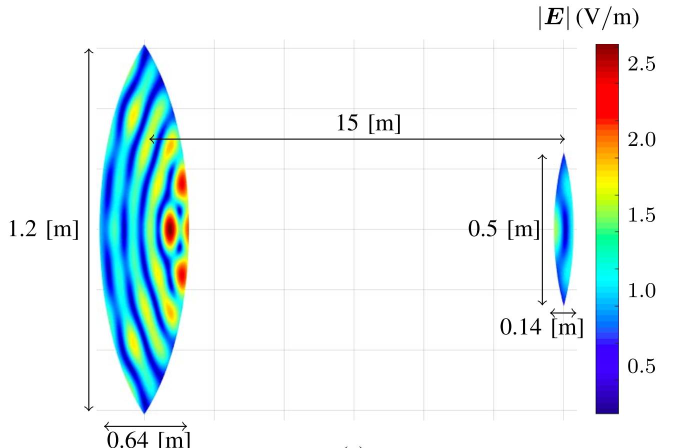


(b)

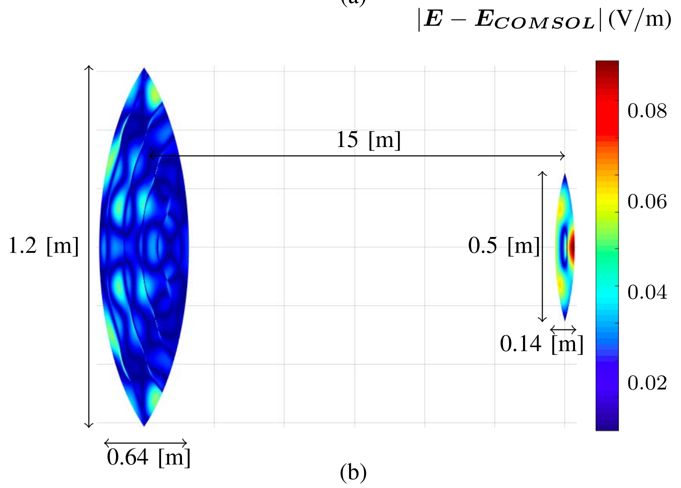
Fig. 10. Memory and CPU time consumption for an exponentially increased number of grid elements at different frequencies. The numbers of triangles in MoM mesh are selected as 1096, 2872, 4322, 7820, and 100 498 for the number of grid points being 4^6, 4^7, 4^8, 4^9, and 4^{10}, respectively, for frequencies of 0.3, 1, and 2.5 GHz. (a) Both memory of storing TT operator (solid line) and peak memory usage (dashed line). (b) Total time of iterative solution (solid line) and the time of TTVM per iteration (dashed line). Both (a) and (b) are presented in logarithmic scale.

in ranks with growing frequency appears to be more significant in the timing complexity, which scales as $O(r_{\max}^2 N \log(N))$. The time per iteration (dashed line) is obtained by dividing total online time by the number of iterations and shows similar behavior as the total time.

The performance of sparsity-aware P-TT in the case of full-wave scattering problems is studied on the mock-up example of two lenses shown in Fig. 11. The case of lenses separated with 15-m distance is considered. Both lenses have relative permittivity $\epsilon_r = 4$. Excitations with the plane wave of 1-V/m magnitude impinging from 0° direction is modeled at 0.6 GHz. The number of unstructured triangle mesh elements and regular grid points are fixed at 1 106 000 and 268 435 456, respectively. In order to get more accurate results in P-TT, five layers of near neighbors were taken in this experiment. The simulated field is shown in Fig. 11(a). The absolute error distribution compared to the finite-element solution of the

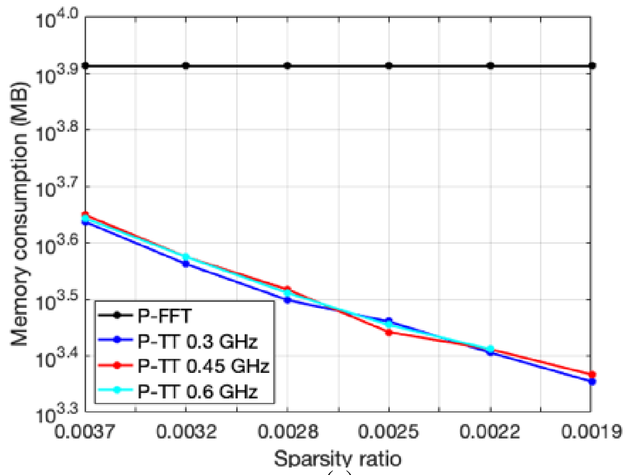


(a)

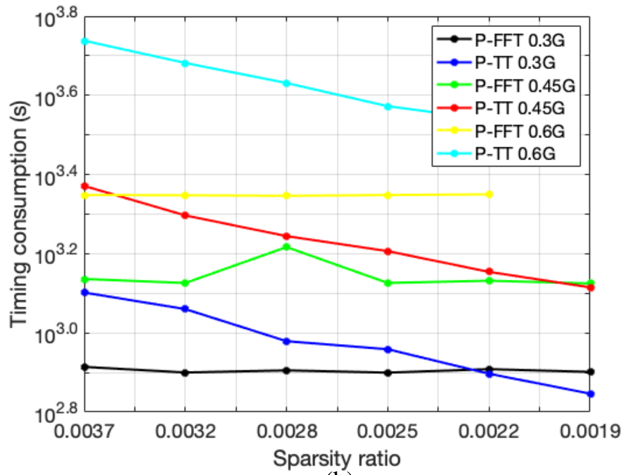


(b)

Fig. 11. (a) Magnitude of electric field in a 2-lens mock up scatterer under 0.6 GHz incident field, and the lenses have 15-m separation distance between their centers. (b) Distribution of absolute error compared with COMSOL.



(a)



(b)

Fig. 12. (a) Peak memory consumptions between sparsity-aware P-TT for 0.3 GHz (blue), 0.45 GHz (red), and 0.6 GHz (cyan) over the sparsity ratios ranging from 0.0019 to 0.0037. P-FFT [10] (black) shows the same memory use for all frequencies and sparsity ratios as expected. The peak total memory usages include storage of $\mathbf{L}^{\text{near}}$, $\hat{\mathcal{L}}^{\text{near}}$, and Λ. (b) CPU time use in P-FFT [10] and P-TT algorithms for the same sparsity ratio values and the same frequencies of 0.3, 0.45, and 0.6 GHz.

commercial COMSOL solver is shown in Fig. 11(b). In this case, the maximum absolute error is 0.086 V/m.

To study the sparsity effect, both P-TT and P-FFT [10] solutions were obtained in the cases of separations increasing from 12 to 17 m, which produced sparsity ratios ranging from 0.0037 down to 0.0019. The experiments were conducted at the frequencies of 0.3, 0.45, and 0.6 GHz. Fig. 12(a) (top) shows the total memory usage for P-FFT and P-TT algorithms, including storage of $\mathbf{L}^{\text{near}}$, $\hat{\mathcal{L}}^{\text{near}}$, and Λ matrices. The number of iterations in the three cases' sparsity remained the same. Therefore, in the case of P-FFT algorithm [10] [black in Fig. 12(a)] for which the memory use does not depend on either increase in frequency or sparsity of grid occupation, the same peak memory is observed. On the other hand, the sparsity-aware P-TT [blue, red, and cyan curves in Fig. 12(a)] shows strong dependence on the sparsity ratio. At all considered sparsity ratios, the P-TT is observed to consume less memory compared with P-FFT [10] in 0.3, 0.45, and 0.6 GHz experiments with 395 800 element meshes. In all the experiments, P-TT consumed about 2.3 GB of memory at the sparsity ratio of 0.0019 and 4.5 GB at the sparsity ratio of 0.0037. This signifies near linear scaling with sparsity as the memory approximately doubles upon doubling in of the sparsity ratio.

Fig. 12(b) shows scaling of CPU time in P-FFT [10] and P-TT algorithms with sparsity ratio and the frequency. As one can see, at 0.3 GHz, the crossover point in the sparsity ratio where P-TT starts outperforming P-FFT is approximately at 0.0022. This crossover point is shifted to the right as frequency increases (0.45 GHz), which shows that for P-TT to outperform P-FFT at a higher frequency, it requires a higher sparsity. The crossover point continues to shift to the right (toward higher sparsity) as frequency increases, hence demonstrating faster increase in CPU time cost in P-TT than P-FFT as frequency grows and higher sparsity required to compensate for it. This is due to the fact that the maximum rank $r_{\max}$ of the TT cores enters into the CPU time of TTVM and memory use as $r_{\max}^2\, O(N \log N)$ and $r_{\max} O(N)$, respectively. Since maximum rank $r_{\max}$ scales as $O(\sqrt{N})$, when frequency grows and N grows respectively, the CPU time and memory complexities for electrically large problems in P-TT scale as $O(N^2 \log N)$ and $O(N^{1.5})$. At the same time, CPU time and memory complexities for P-FFT scale only as $O(N \log N)$. However, while P-FFT cannot account for sparsity of the scatterer, the P-TT can. As a result, the proposed P-TT algorithm is more favorable for the solution of geometrically complex but electrically moderate in size problems (up to several wavelengths) where it can outperform classical P-FFT when sufficient sparsity is present in the model.

VII. Conclusion

A sparsity-aware P-TT iterative fast algorithm is proposed for the solution of static and full-wave problems in 2-D. The method enables the use of TT decomposition of the Toeplitz matrices for the acceleration of MoM solution on unstructured meshes of VIE. Surface integral equations of electromagnetics can be analogously handled by the proposed P-TT algorithm. The concept of projection of the sources from unstructured meshes onto point sources of regular Cartesian grids is utilized by analogy with the well-established P-FFT [10] algorithm to decouple the MoM mesh from the regular grid required for organizing the MoM interactions into a Toeplitz matrix. The TT decomposition compresses the generator of the Toeplitz matrix, which leads to substantial memory reduction in the P-TT approach compared to the P-FFT [10] algorithm. Also, unlike the P-FFT [10] method, the P-TT allows for further reduction of CPU time and memory by accounting for the sparsity pattern in the occupation of the regular Cartesian grid by the object of interest. In this article, we demonstrate how such sparsity can be accounted for in multiplication of the cores of the TT-decomposed Toeplitz matrix with a vector. It is shown that upon sufficient sparsity, the P-TT algorithm can outperform P-FFT [10] in CPU time as well as memory use. Presented ideas will be more beneficial in the future 3-D extensions of this article as the sparsity ratios are significantly larger in realistic 3-D structures, especially when acceleration of 3-D surface integral equations is considered.

References

[1] N. N. Bojarski, “K-space formulation of the electromagnetic scattering problems,” Air Force Avionic Lab., Wright-Patterson Air Force Base, OH, USA, Tech. Rep. AFAL-TR-71-75, Mar. 1971.

[2] M. F. Catedra, R. F. Torres, J. Basterrechea, and E. Gago, *The CG-FFT Method: Application of Signal Processing Techniques to Electromagnetics*. Boston, MA, USA: Artech House, 1995.

[3] A. W. Appel, “An efficient program for many-body simulation,” *SIAM J. Sci. Stat. Comput.*, vol. 6, no. 1, pp. 85–103, Jan. 1985.

[4] J. Barnes and P. Hut, “A hierarchical O(N log N) force-calculation algorithm,” *Nature*, vol. 324, pp. 446–449, Dec. 1986.

[5] L. Greengard and V. Rokhlin, “A fast algorithm for particle simulations,” *J. Comput. Phys.*, vol. 73, no. 2, pp. 325–348, Dec. 1987.

[6] V. Rokhlin, “Rapid solution of integral equations of scattering theory in two dimensions,” *J. Comput. Phys.*, vol. 36, no. 2, pp. 414–439, 1990.

[7] R. Coifman, V. Rokhlin, and S. Wandzura, “The fast multipole method for the wave equation: A pedestrian prescription,” *IEEE Antennas Propag. Mag.*, vol. 35, no. 3, pp. 7–12, Jun. 1993.

[8] T. J. Cui and W. C. Chew, “Fast algorithm for electromagnetic scattering by buried 3-D dielectric objects of large size,” *IEEE Trans. Geosci. Remote Sens.*, vol. 37, no. 5, pp. 2597–2608, Sep. 1999.

[9] C.-F. Wang, F. Ling, and J.-M. Jin, “A fast full-wave analysis of scattering and radiation from large finite arrays of microstrip antennas,” *IEEE Trans. Antennas Propag.*, vol. 46, no. 10, pp. 1467–1474, Oct. 1998.

[10] J. R. Phillips and J. White, “A precorrected-FFT method for capacitance extraction of complicated 3-D structures,” in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des.*, Nov. 1994, pp. 268–271.

[11] J. R. Phillips and J. K. White, “Efficient capacitance extraction of 3D structures using generalized pre-corrected FFT methods,” in *Proc. IEEE 3rd Topical Meeting Elect. Perform. Electron. Packag.*, Nov. 1994, pp. 253–256.

[12] E. Bleszynski, M. Bleszynski, and T. Jaroszewicz, “A fast integral-equation solver for electromagnetic scattering problems,” in *Antennas Propag. Soc. Int. Symp. (AP-S) Dig.*, vol. 1, Jun. 1994, pp. 416–419.

[13] E. Bleszynski, M. Bleszynski, and T. Jaroszewicz, “AIM: Adaptive integral method for solving large-scale electromagnetic scattering and radiation problems,” *Radio Sci.*, vol. 31, no. 5, pp. 1225–1251, Sep. 1996.

[14] W. Hackbusch, “A sparse matrix arithmetic based on $\mathcal{H}$-matrices. Part I: Introduction to $\mathcal{H}$-matrices,” *Computing*, vol. 62, no. 2, pp. 89–108, 1999.

[15] H. A. van der Vorst, “Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems,” *SIAM J. Sci. Stat. Comput.*, vol. 13, no. 2, pp. 631–644, 1992.

[16] Y. Saad and M. H. Schultz, “GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems,” *SIAM J. Sci. Stat. Comput.*, vol. 7, no. 3, pp. 856–869, Jul. 1986.

[17] H. Guo, J. Hu, and E. Michielssen, “On MLMDA/butterfly compressibility of inverse integral operators,” *IEEE Antennas Wireless Propag. Lett.*, vol. 12, pp. 31–34, 2013.

[18] S. Omar and D. Jiao, “A linear complexity direct volume integral equation solver for full-wave 3-D circuit extraction in inhomogeneous materials,” *IEEE Trans. Microw. Theory Techn.*, vol. 63, no. 3, pp. 897–912, Mar. 2015.

[19] I. Oseledets, “Compact matrix form of the D-dimensional tensor decomposition,” in *Proc. INM RAS*, Mar. 2009, pp. 1–3.

[20] I. Oseledets and E. Tyrtyshnikov, “TT-cross approximation for multidimensional arrays,” *Linear Algebra Appl.*, vol. 432, no. 1, pp. 70–88, Jan. 2010.

[21] I. Oseledets. (2012). *TT-Toolbox 2.2*. [Online]. Available: https://github.com/oseledets/TT-Toolbox

[22] Z. Chen, S. Zheng, and V. I. Okhmatovski, “Tensor train accelerated solution of volume integral equation for 2-D scattering problems and magneto-quasi-static characterization of multiconductor transmission lines,” *IEEE Trans. Microw. Theory Techn.*, to be published.

[23] A. C. Yucel, I. P. Georgakis, A. G. Polimeridis, H. Bağci, and J. K. White, “VoxHenry: FFT-accelerated inductance extraction for voxelized geometries,” *IEEE Trans. Microw. Theory Techn.*, vol. 66, no. 4, pp. 1723–1735, Apr. 2018.

[24] J. D. Morsey, V. I. Okhmatovski, and A. C. Cangellaris, “Finite-thickness conductor models for full-wave analysis of interconnects with a fast integral equation method,” *IEEE Trans. Adv. Packag.*, vol. 27, no. 1, pp. 24–33, Feb. 2004.

[25] Z. Chen, S. Zheng, Q. Cheng, A. Yucel, and V. I. Okhmatovski, “Pre-corrected tensor train algorithm for current flow modelling in 2D multi-conductor transmission lines,” in *IEEE MTT-S Int. Microw. Symp. Dig.*, Boston, MA, USA, Jun. 2019, pp. 124–127.

[26] W. C. Chew, *Waves and Fields in Inhomogeneous Media*. Piscataway, NJ, USA: IEEE Press, 1999.

[27] F. R. Gantmacher, *Theory of Matrices*. New York, NY, USA: Chelsea, 1959.

[28] S. Omar, M. Ma, and D. Jiao, “Low-complexity direct and iterative volume integral equation solvers with a minimal-rank $\mathcal{H}^2$-representation for large-scale three-dimensional electrodynamic analysis,” *IEEE J. Multiscale Multiphys. Comput. Techn.*, vol. 2, pp. 210–223, 2017.

[29] E. Corona, A. Rahimian, and D. Zorin, “A tensor-train accelerated solver for integral equations in complex geometries,” *J. Comput. Phys.*, vol. 334, pp. 145–169, Apr. 2017.

[30] M. Shafieipour, Z. Chen, A. Menshov, J. De Silva, and V. I. Okhmatovski, “Efficiently computing the electrical parameters of cables with arbitrary cross-sections using the method-of-moments,” *Electr. Power Syst. Res.*, vol. 162, pp. 37–49, Sep. 2018.

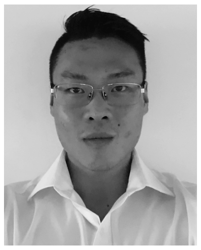
[31] W. C. Chew, *Waves and Fields in Inhomogeneous Media*. Piscataway, NJ, USA: IEEE Press, 1999.

[32] J. Morsey, “Integral equation methodologies for the signal integrity analysis of PCB and interconnect structures in layered media from DC to multi-GHz frequencies,” Ph.D. dissertation, Dept. Elect. Comput. Eng., Univ. Illinois Urbana-Champaign, Champaign, IL, USA, 2004.

[33] M. Kamon, M. J. Tsuk, and J. K. White, “FASTHENRY: A multipole-accelerated 3-D inductance extraction program,” *IEEE Trans. Microw. Theory Techn.*, vol. 42, no. 9, pp. 1750–1758, Sep. 1994.

[34] J. R. Phillips and J. K. White, “A precorrected-FFT method for electrostatic analysis of complicated 3-D structures,” *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 16, no. 10, pp. 1059–1072, Oct. 1997.

[35] V. I. Okhmatovski, M. Yuan, I. Jeffrey, and R. Phelps, "A three-dimensional precorrected FFT algorithm for fast method of moments solutions of the mixed-potential integral equation in layered media," *IEEE Trans. Microw. Theory Techn.*, vol. 57, no. 12, pp. 3505–3517, Dec. 2009.



Zhuotong Chen (S'17) received the B.S. degree (Hons.) in electrical and computer engineering from the University of Manitoba, Winnipeg, MB, Canada, in 2018. He is currently pursuing the Ph.D. degree in electrical and computer engineering at the University of California at Santa Barbara (UCSB), Santa Barbara, CA, USA.

From 2017 to 2018, he was a Research Assistant with the University of Manitoba focusing on fast algorithms for computational electromagnetics. His current research interests include tensor methods for computational electromagnetics, uncertainty quantification with application for MRI simulation, and numerical optimization for data analysis.

Luis J. Gomez (M'08) received the B.S. degree in electrical engineering and mathematics from the University of Florida, Gainesville, FL, USA, in 2008, and the M.S. degree in electrical engineering and applied mathematics and the Ph.D. degree in electrical engineering from the University of Michigan, Ann Arbor, MI, USA, in 2014 and 2015, respectively.

He is currently a Post-Doctoral Fellow with the Department of Psychiatry and Behavioral Sciences, Duke University School of Medicine, Durham, NC, USA, where he is currently involved in the development of optimization and computational techniques for use in improving noninvasive brain stimulation procedures. Previously, he was a Post-Doctoral Fellow with the Radiation Laboratory, University of Michigan at Ann Arbor from 2015 to 2016, where he developed fast-integral equation methods for analyzing scattering by highly heterogeneous media and inverse scattering methods. His main research interests include computational electromagnetism, with a focus on wave propagation in highly heterogeneous materials, volume integral equations, electromagnetic imaging and optimization, and uncertainty quantification of biomedical applications.

Dr. Gomez was a recipient of the National Science Foundation Graduate Fellowship in 2008 and the National Institutes of Health BRAIN Initiative K99/R00 Post-Doctoral Training Award in 2019.

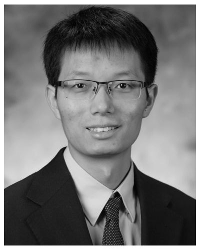
Abdulkadir C. Yucel (M'19-SM'19) received the B.S. degree in electronics engineering from the Gebze Institute of Technology, Kocaeli, Turkey, in 2005, and the M.S. and Ph.D. degrees in electrical engineering from the University of Michigan, Ann Arbor, MI, USA, in 2008 and 2013, respectively.

From September 2005 to August 2006, he was a Research and Teaching Assistant with the Gebze Institute of Technology. From August 2006 to April 2013, he was a Graduate Student Research Assistant with the University of Michigan. From May 2013 to December 2017, he was a Post-Doctoral Research Fellow with the University of Michigan, the Massachusetts Institute of Technology, Cambridge, MA, USA, and the King Abdullah University of Science and Technology, Thuwal, Saudi Arabia. Since 2018, he has been an Assistant Professor with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore. His current research interests include various aspects of computational electromagnetics with an emphasis on uncertainty quantification for electromagnetic analysis on complex platforms, electromagnetic compatibility and interference analysis, nature-based design of electromagnetic devices, and integral equation-based frequency and time-domain solvers and their accelerators.

Dr. Yucel was a recipient of the Fulbright Fellowship in 2006, the Electrical Engineering and Computer Science Departmental Fellowship of the University of Michigan in 2007, and the Student Paper Competition Honourable Mention Award at the IEEE International Symposium on Antennas and the Propagation Symposium in 2009. He has been serving as an Associate Editor for the *International Journal of Numerical Modelling: Electronic Networks, Devices and Fields* and as a reviewer for various technical journals.

Shucheng Zheng (S'16) was born in Fujian, China, in 1987. He received the B.Sc. degree (Hons.) in electrical and computer engineering from the University of Manitoba, Winnipeg, MB, Canada, in 2016, where he is currently pursuing the Ph.D. degree in electrical and computer engineering.

His current research interests include computational electromagnetics, multilayered media Green's functions, high-performance computing, modeling of high-speed interconnects, and transient analysis of power systems.



Zheng Zhang (M'15) received the Ph.D. degree in electrical engineering and computer science from the Massachusetts Institute of Technology (MIT), Cambridge, MA, USA, in 2015.

He has been an Assistant Professor of electrical and computer engineering with the University of California at Santa Barbara (UCSB), Santa Barbara, CA, USA, since 2017. His industrial experiences include Coventor Inc., Raleigh, NC, USA and Maxim-IC, San Jose, CA, USA; academic visiting experiences include UC San Diego, La Jolla, CA, USA, Brown University, Providence, RI, USA, and Politechnico di Milano, Milan, Italy; and government lab experiences include Argonne National Laboratory, Lemont, IL, USA. His current research interests include uncertainty quantification with applications to the design automation of multidomain systems (e.g., nanoscale electronics, integrated photonics, and autonomous systems), and tensor computational methods for high-dimensional data analytics.

Dr. Zhang was a recipient of the Best Paper Award of the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS in 2014, the Best Paper Award of the IEEE TRANSACTIONS ON COMPONENTS, PACKAGING AND MANUFACTURING TECHNOLOGY in 2018, two Best Paper Awards at the IEEE EPEPS 2018 and the IEEE SPI 2016, and three additional Best Paper Nominations at international conferences, such as the CICC 2014, ICCAD 2011, and ASP-DAC 2011. His Ph.D. dissertation was recognized by the ACM SIGDA Outstanding Ph.D. Dissertation Award in Electronic Design Automation in 2016 and by the Doctoral Dissertation Seminar Award (i.e., Best Thesis Award) from the Microsystems Technology Laboratory, MIT in 2015. He was also a recipient of the Li Ka-Shing Prize from the University of Hong Kong in 2011 and the NSF CAREER Award in 2019.

Vladimir I. Okhmatovski (M'99–SM'09) was born in Moscow, Russia, in 1974. He received the M.S. degree (Hons.) in radiophysics and the Ph.D. degree in antennas and microwave circuits from the Moscow Power Engineering Institute, Moscow, in 1996 and 1997, respectively.

In 1997, he joined the Radio Engineering Department, Moscow Power Engineering Institute, as an Assistant Professor. He was a Post-Doctoral Research Associate with the National Technical University of Athens, Athens, Greece, from 1998 to 1999, and with the University of Illinois at Urbana–Champaign, Champaign, IL, USA, from 1999 to 2003. From 2003 to 2004, he was with the Department of Custom Integrated Circuits, Cadence Design Systems, San Jose, CA, USA, as a Senior Member of Technical Staff, and from 2004 to 2008 as an Independent Consultant. In 2004, he joined the Department of Electrical and Computer Engineering, University of Manitoba, Winnipeg, MB, Canada, where is currently a Full Professor. His current research interests include fast algorithms of electromagnetics, high-performance computing, modeling of interconnects, and inverse problems.

Dr. Okhmatovski was a recipient of the 2017 Intel Corporate Research Council Outstanding Researcher Award, the 1995 Scholarship of the Government of Russian Federation, the 1996 Scholarship of the President of the Russian Federation, and the 1996 Best Young Scientist Report of the VI International Conference on Mathematical Methods in Electromagnetic Theory. He was also a co-recipient of the Best Paper Award of the 3rd Electronic Packaging Technology Conference in 2001 and the Outstanding ACES Journal Paper Award in 2007. He is a Registered Professional Engineer in the Province of Manitoba, Canada.