Communication

A Memory-Efficient Implementation of Perfectly Matched Layer With Smoothly Varying Coefficients in Discontinuous Galerkin Time-Domain Method

Liang Chen, Mehmet Burak Özakın, Shehab Ahmed, and Hakan Bagci

***Abstract*— Wrapping a computation domain with a perfectly matched layer (PML) is one of the most effective methods of imitating/approximating the radiation boundary condition in the Maxwell and wave equation solvers. Many PML implementations often use a smoothly increasing attenuation coefficient to increase the absorption for a given layer thickness, and, at the same time, to reduce the numerical reflection from the interface between the computation domain and the PML. In discontinuous Galerkin time-domain (DGTD) methods, using a PML coefficient that varies within a mesh element requires a different mass matrix to be stored for every element and therefore significantly increases the memory footprint. In this work, this bottleneck is addressed by applying a weight-adjusted approximation to these mass matrices. The resulting DGTD scheme has the same advantages as the scheme that stores individual mass matrices, namely, higher accuracy (due to reduced numerical reflection) and increased meshing flexibility (since the PML does not have to be defined layer by layer), but it requires significantly less memory.**

***Index Terms*— Absorbing boundary conditions, discontinuous Galerkin (DG) method, perfectly matched layer (PML), time-domain analysis, weight-adjusted approximation (WAA).**

I. Introduction

Perfectly matched layer (PML) [1], [2] is often used in finite difference [3], finite element [4], and discontinuous Galerkin time-domain (DGTD) methods [5]–[10] to imitate/approximate the radiation boundary condition (i.e., truncate an unbounded physical domain to a finite computation domain) while solving Maxwell equations or the wave equation.

The theoretical performance of the PML (in the continuous space) depends on the attenuation coefficient (which is implemented as conductivity in Maxwell equations) and the thickness of the layer [1]. Increasing either one or both of them increases the absorption inside the PML. However, in practice, the PML is discretized using mesh elements, which results in numerical reflections [1], [11]. One can use a constant and high conductivity, but this increases the numerical reflection at the interface between the PML and the computation domain. One can also use a very thick layer (or many mesh elements), but this increases the computational cost. A solution to these problems is to use a smoothly increasing conductivity profile that yields both high absorption and small numerical reflection [1]–[4], [11]–[14].

In DGTD, the PML conductivity profile can be implemented in two different ways. The first method assumes that the conductivity is constant in a given element. This constant might, for example, be set to the value of the conductivity profile at the center of the element. Implementation of this method is rather straightforward since the mass matrices of different elements only differ by a constant (for linear elements) [8]. However, the conductivity profile becomes discontinuous between the neighboring elements [see Fig. 1(a)], and the element surfaces that are not parallel to the PML interface lead to large reflections and destroy the high-order accuracy of the solution. One workaround is to build layered (tetrahedral) meshes [see Fig. 1(b)] or use orthogonal (hybrid) meshes inside the PML [7], [15] and accordingly set a layered conductivity profile. However, this makes the setups of the computation domain and the PML rather tedious since one needs to control the mesh and the conductivity on all face, edge, and corner regions of the PML. Moreover, to reduce the numerical reflection by decreasing the conductivity discontinuity between the neighboring mesh (or conductivity) layers, their number has to be increased.

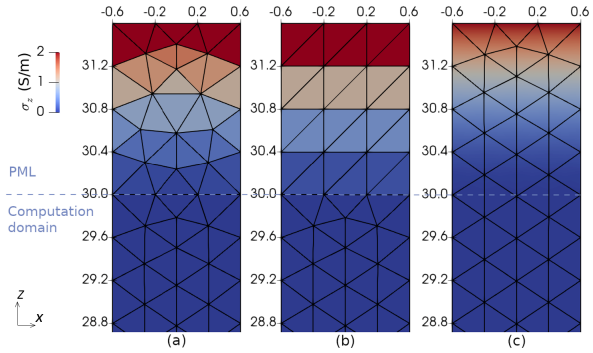

Fig. 1. Implementation of the PML with smoothly increasing conductivity. (a) Conductivity is constant in elements (paved mesh). (b) Conductivity is constant in elements (layered mesh). (c) Conductivity is allowed to vary in elements (paved mesh). Mesh does not have to conform to the interface between the PML and the computation domain.

The second method allows the conductivity to vary inside a given element (following the increasing conductivity profile inside the PML). It should be noted here that (higher-order) DGTD allows for sampling of the material properties at the subelemental level. In this case, the behavior of the PML is determined only by the conductivity samples within the elements, and therefore, the mesh interfaces can be aligned arbitrarily [see Fig. 1(c)] without adversely affecting its performance. Indeed, this second approach can improve the PML performance [16]–[18]. However, the conductivity varying within the elements results in an element-dependent mass matrix. Therefore, a direct implementation requires a different mass matrix (or its inverse) to be stored for every element [7], [9], [15]–[17], [19]. This significantly increases DGTD's memory footprint. For example, for the stretched-coordinate (SC)-PML [9], the memory required to store the mass matrices scales with $15K_{\text{PML}} \times N_p^2$, where N_p is the number of interpolating nodes in each element, K_{PML} is the number of elements in the PML, and 15 comes from the five material-dependent coefficients in the update equations and

Manuscript received July 10, 2020; revised October 6, 2020; accepted October 25, 2020. Date of publication November 18, 2020; date of current version June 2, 2021. This work was supported in part by the King Abdullah University of Science and Technology (KAUST) Office of Sponsored Research (OSR) under Award 2019-CRG8-4056 and in part by the KAUST Ali I. Al-Naimi Petroleum Engineering Research Center (ANPERC). *(Corresponding author: Liang Chen.)*

The authors are with the Division of Computer, Electrical, and Mathematical Science and Engineering, King Abdullah University of Science and Technology (KAUST), Thuwal 23955-6900, Saudi Arabia (e-mail: liang.chen@kaust.edu.sa; mehmet.ozakin@kaust.edu.sa; shehab.ahmed@kaust.edu.sa; hakan.bagci@kaust.edu.sa).

Color versions of one or more figures in this communication are available at https://doi.org/10.1109/TAP.2020.3037651.

Digital Object Identifier 10.1109/TAP.2020.3037651

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

three Cartesian components of the vector field. In contrast, for the first method, where the conductivity is assumed constant in a given element, this memory requirement scales with K_{PML} since only the constant conductivity of each element and a single reference mass matrix are stored.

In this work, a memory-efficient method to implement the SC-PML with smoothly varying attenuation coefficients in DGTD is developed. The proposed method allows the conductivity to vary inside the elements and constructs the resulting local mass matrices using a weight-adjusted approximation (WAA) [20]. Compared with the direct implementation that is briefly described earlier, the proposed method reduces the memory requirement to $15K_{\text{PML}} \times N_q$, where $N_q \sim N_p$, while maintaining the PML performance. It should be noted here that the WAA has been proved to be energy-stable and preserve the high-order convergence of DGTD [20]–[22]. Indeed, numerical examples presented here also show that the proposed method maintains the higher order accuracy of the solution. In addition, it is demonstrated that the PML with smoothly increasing conductivity profile as implemented with the proposed method performs better than the PML implemented using an elementwise constant conductivity profile.

II. Formulation

A. WAA-DGTD for SC-PML

Maxwell equations in SCs for a source-free and lossless medium can be expressed as [2]

$$-j\omega\mu\mathbf{H} = \nabla_e \times \mathbf{E} \tag{1}$$

$$j\omega\epsilon\mathbf{E} = \nabla_h \times \mathbf{H} \tag{2}$$

where $\mathbf{E}$ and $\mathbf{H}$ are the electric and magnetic fields, respectively, ϵ and μ are the permittivity and the permeability, respectively, ω is the frequency, and

$$\nabla_e = \nabla_h = \hat{x}\frac{1}{s_x}\frac{\partial}{\partial x} + \hat{y}\frac{1}{s_y}\frac{\partial}{\partial y} + \hat{z}\frac{1}{s_z}\frac{\partial}{\partial z}. \tag{3}$$

The coordinate-stretching variables s_u, $u \in \{x, y, z\}$, are defined as [2], [9], [14]

$$s_u(u) = \kappa_u(u) + \frac{\sigma_u(u)}{j\omega\varepsilon_0} \tag{4}$$

where κ_u and σ_u are 1-D positive real functions along direction u. Here, σ_u is the attenuation coefficient that ensures absorption inside the SC-PML and κ_u changes the propagation speed (and also increases the absorption for evanescent waves). It is well known that using smoothly increasing σ_u and κ_u reduces the numerical reflection from the interface between the PML and the computation domain while maintaining a high absorption rate and therefore improves the PML performance [11], [13], [14].

The time-domain update equations in SC-CPML can be expressed as [9]

$$\partial_t \ddot{a} \cdot \mu\mathbf{H} = -\nabla \times \mathbf{E} - \ddot{b} \cdot \mu\mathbf{H} - \ddot{c} \cdot \mu\mathbf{P}^H \tag{5}$$

$$\partial_t \ddot{a} \cdot \varepsilon\mathbf{E} = \nabla \times \mathbf{H} - \ddot{b} \cdot \varepsilon\mathbf{E} - \ddot{c} \cdot \mu\mathbf{P}^E \tag{6}$$

$$\partial_t \mathbf{P}^H = \ddot{\kappa}^{-1}\mathbf{H} - \ddot{d}\mathbf{P}^H \tag{7}$$

$$\partial_t \mathbf{P}^E = \ddot{\kappa}^{-1}\mathbf{E} - \ddot{d}\mathbf{P}^E \tag{8}$$

where $\mathbf{P}^E$ and $\mathbf{P}^H$ are auxiliary variables introduced to avoid computationally costly temporal convolutions while converting (1) and (2) into time domain [9] and $\ddot{a}$, $\ddot{b}$, $\ddot{c}$, $\ddot{d}$, and $\ddot{\kappa}$ are diagonal tensors with entries defined as

$$a_{uu} = \frac{\kappa_v\kappa_w}{\kappa_u}, \quad b_{uu} = \frac{1}{\kappa_u\varepsilon_0}(\sigma_v\kappa_w + \sigma_w\kappa_v - a_{uu}\sigma_u)$$
$$c_{uu} = \frac{\sigma_v\sigma_w}{\varepsilon_0^2} - b_{uu}\frac{\sigma_u}{\varepsilon_0}, \quad d_{uu} = \frac{\sigma_u}{\kappa_u\varepsilon_0}, \quad \kappa_{uu} = \kappa_u. \tag{9}$$

In (9), the rest of the text (u, v, w) follows the permutation $(x, y, z) \rightarrow (y, z, x) \rightarrow (z, x, y)$.

Following the standard nodal discontinuous Galerkin (DG) method [8], the computation domain and the PML are discretized into K elements with volumetric support Ω_k and boundary surface $\partial\Omega_k$, and in each element, $\mathbf{E}$, $\mathbf{H}$, $\mathbf{P}^E$, and $\mathbf{P}^H$ are expanded using the Lagrange polynomials $\ell_i(\mathbf{r})$ [8], $i = 1, \ldots, N_p$, where $N_p = (p+1)(p+2)(p+3)/6$ is the number of interpolating nodes and p is the order of the Lagrange polynomials. Finally, Galerkin testing yields the semidiscrete system of equations as

$$\partial_t \bar{H}_k = -\left(\bar{M}_k^a\right)^{-1}\left[\bar{M}_k^b\bar{H}_k + \bar{M}_k^c\bar{P}_k^H + \mu_k^{-1}\bar{\mathbb{C}}_k(\bar{E}_k, \bar{E}_{k'}, \bar{H}_k, \bar{H}_{k'})\right] \tag{10}$$

$$\partial_t \bar{E}_k = -\left(\bar{M}_k^a\right)^{-1}\left[\bar{M}_k^b\bar{E}_k + \bar{M}_k^c\bar{P}_k^E - \varepsilon_k^{-1}\bar{\mathbb{C}}_k(\bar{H}_k, \bar{H}_{k'}, \bar{E}_k, \bar{E}_{k'})\right] \tag{11}$$

$$\partial_t \bar{P}_k^H = \bar{M}_k^{-1}\left(\bar{M}_k^{1/\kappa}\bar{H}_k - \bar{M}_k^d\bar{P}_k^H\right) \tag{12}$$

$$\partial_t \bar{P}_k^E = \bar{M}_k^{-1}\left(\bar{M}_k^{1/\kappa}\bar{E}_k - \bar{M}_k^d\bar{P}_k^E\right). \tag{13}$$

where $\bar{H}_k$, $\bar{E}_k$, $\bar{P}_k^H$, and $\bar{P}_k^E$ are vectors storing the unknown time-dependent coefficients of the relevant basis functions, $\bar{M}_k$ and $\bar{M}_k^\alpha$, $\alpha \in \{a, b, c, d, 1/\kappa\}$, are the mass matrices with entries

$$\bar{M}_k(i, j) = \int_{\Omega_k} \ell_i(\mathbf{r})\ell_j(\mathbf{r})d\mathbf{r} \tag{14}$$

$$\bar{M}_k^{\alpha,u}(i, j) = \int_{\Omega_k} \alpha_{uu}(\mathbf{r})\ell_i(\mathbf{r})\ell_j(\mathbf{r})d\mathbf{r} \tag{15}$$

$\bar{\mathbb{C}}_k(f_k, f_{k'}, g_k, g_{k'})$ denotes the curl operator with its component along direction u, which is given by

$$\bar{\mathbb{C}}_k^u(f_k, f_{k'}, g_k, g_{k'}) = \bar{S}_k^v f_k^w - \bar{S}_k^w f_k^v + \bar{F}_k\mathbb{F}^u(f_k, f_{k'}, g_k, g_{k'})$$

where $u \in \{x, y, z\}$, $(f, g) \in \{(\bar{E}, \bar{H}), (\bar{H}, \bar{E})\}$, $\mathbb{F}^u$ is the component of the numerical flux along direction u, which in general involves unknowns from the current element k and its neighboring element k' [8], [23], [24], $\bar{S}_k$ and $\bar{F}_k$ are the stiffness and the face mass matrices with entries, respectively

$$\bar{S}_k^u(i, j) = \int_{\Omega_k} \ell_i(\mathbf{r})\frac{d\ell_j(\mathbf{r})}{du}d\mathbf{r} \tag{16}$$

$$\bar{F}_k(i, j) = \oint_{\partial\Omega_k} \ell_i(\mathbf{r})\ell_j(\mathbf{r})d\mathbf{r} \tag{17}$$

and ϵ_k and μ_k are the permittivity and the permeability (assumed constant) in each element, respectively. Note that, the nodal DG framework [8] is used here, but the proposed method can be easily extended to vector DG methods [6], [7], [9], [25]–[27].

For linear elements, the mass matrices $\bar{M}_k$ in (14) are simply scaled versions of the mass matrix $\bar{M}$ defined on the reference element, $\bar{M}_k = J_k\bar{M}$, where J_k is the Jacobian of the coordinate transformation between element k and the reference element. Hence, only $\bar{M}$ and (scalar constant) J_k are stored. Similarly, in (15), if $\alpha_{uu}(\mathbf{r})$ is assumed constant inside the elements, i.e., $\alpha_{uu}(\mathbf{r}) = \alpha_{uu}^k$, then $\bar{M}_k^{\alpha,u} = \alpha_{uu}^k\bar{M}_k = \alpha_{uu}^k J_k\bar{M}$ and one only needs to store α_{uu}^k in addition to $\bar{M}$ and J_k. In this case, (10)–(13) can be implemented as efficiently as the case without the PML [8], [28], [29].

However, if $\alpha_{uu}(\mathbf{r})$ is allowed to vary inside the elements, $\bar{M}_k^{\alpha,u}$ are different in different elements, and in general, there is no simple relationship between these different mass matrices. One has to store every one of these mass matrices (or their inverse). As an alternative, the mass matrix can be recomputed at each time step, but this would significantly increase the cost of time marching [8]. The memory required to store the mass matrices $\bar{M}_k^{\alpha,u}$ in (10)–(13) scales with $3 \times 5 \times N_p^2$ per element, where 3 is the number of the (x, y, z)

components of the vector field, and 5 is the number of the coefficients $a(\mathbf{r})$, $b(\mathbf{r})$, $c(\mathbf{r})$, $d(\mathbf{r})$, and $\kappa(\mathbf{r})$. Note that this memory requirement is significantly higher than that of storing the unknowns coefficients of the basis functions, which scales with $12 \times N_p$ in the PML.

To reduce the memory requirement of implementing (15) with $\alpha_{uu}(\mathbf{r})$ allowed to vary inside the elements, WAA [20] is used. It has been shown that with this approximation, DGTD retains provable energy-stability and high-order accuracy [20]–[22]. Note that in the above SC-PML formulation, directly multiplying (5) and (6) with $\ddot{a}^{-1}$ on both sides reduces the number of element-dependent mass matrices to 4. However, this would result in a nonconservative form, whose solution is neither provably energy-stable nor provably high-order accurate [8], [20].

First, a weight-adjusted inner product is introduced to approximate the parameter-weighted inner product in the expression of the mass matrix [20]. The mass matrix, which is associated with the element k and a locally varying coefficient $\alpha_k(\mathbf{r})$, is approximated as

$$\bar{M}_k^{\alpha} \approx \bar{M}_k (\bar{M}_k^{1/\alpha})^{-1} \bar{M}_k. \tag{18}$$

Since $(\bar{M}_k^{\alpha})^{-1}$ is used in (10)–(13) (for $\alpha = a$), one needs to calculate $\bar{M}_k^{1/\alpha}$. Under the nodal DG framework [8]

$$\begin{aligned} \bar{M}_k^{1/\alpha}(i,j) &= J_k \int_{\Omega_k} \alpha_k^{-1}(\mathbf{r}) \ell_i(\mathbf{r}) \ell_j(\mathbf{r}) d\mathbf{r} \\ &\approx J_k \sum_q \ell_i(\mathbf{r}_q) w_q \alpha_k^{-1}(\mathbf{r}_q) \ell_j(\mathbf{r}_q) \end{aligned} \tag{19}$$

where $\mathbf{r}_q$, $q = 1, \ldots, N_q$, are the Gaussian quadrature nodes corresponding to the quadrature rules of degree $2p+1$ and w_q are the corresponding weights. Hence,

$$\bar{M}_k^{1/\alpha} = J_k \bar{V}_q^T \bar{w}_q \bar{\alpha}_k^{-1} \bar{V}_q \tag{20}$$

where $\bar{V}_q$ is an interpolation matrix defined on the reference element, $\bar{V}_q = \bar{V}_I \bar{V}^{-1}$, $\bar{V}_I$ and $\bar{V}$ are the generalized Vandermonde matrices with entries $\bar{V}_I(q,i) = \phi_i(\mathbf{r}_q)$ and $\bar{V}(j,i) = \phi_i(\mathbf{r}_j)$, respectively, $\phi_i(\mathbf{r})$ is the ith orthonormal polynomial basis [8], $\bar{w}_q = \text{diag}\{w_1, \ldots, w_{N_q}\}$ is also element-independent, and $\bar{\alpha}_k = \text{diag}\{\alpha_k(\mathbf{r}_1), \ldots, \alpha_k(\mathbf{r}_{N_q})\}$ is a diagonal matrix containing the coefficients evaluated at the quadrature nodes.

Inverting (18) and substituting (20) yield

$$\begin{aligned} (\bar{M}_k^{\alpha})^{-1} &\approx \bar{M}_k^{-1} \bar{M}_k^{1/\alpha} \bar{M}_k^{-1} \\ &= \bar{M}^{-1} \bar{V}_q^T \bar{w}_q \bar{\alpha}_k^{-1} \bar{V}_q \bar{M}_k^{-1} \\ &= \bar{P}_q \bar{\alpha}_k^{-1} \bar{V}_q \bar{M}_k^{-1} \end{aligned} \tag{21}$$

where $\bar{P}_q = \bar{M}^{-1} \bar{V}_q^T \bar{w}_q$ is introduced to simplify the implementation. In (21), $\bar{P}_q$ and $\bar{V}_q$ are defined on the reference element, and $\bar{M}_k^{-1}$ is a scaled version of the reference matrix, $\bar{M}_k^{-1} = J_k^{-1} \bar{M}^{-1}$.

The update equations (10)–(13) contain multiplications between element-dependent mass matrices. To reduce the number of arithmetic operations, the following operators are defined:

$$\tilde{M}_k^b = \left(\bar{M}_k^a\right)^{-1} \bar{M}_k^b = \bar{P}_q \bar{a}_k^{-1} \bar{V}_q \bar{P}_q \bar{b}_k \bar{V}_q \tag{22}$$

$$\tilde{M}_k^c = \left(\bar{M}_k^a\right)^{-1} \bar{M}_k^c = \bar{P}_q \bar{a}_k^{-1} \bar{V}_q \bar{P}_q \bar{c}_k \bar{V}_q \tag{23}$$

$$\tilde{M}_k^d = \bar{M}_k^{-1} \bar{M}_k^d = \bar{P}_q \bar{d}_k \bar{V}_q \tag{24}$$

$$\tilde{M}_k^{1/\kappa} = \bar{M}_k^{-1} \bar{M}_k^{1/\kappa} = \bar{P}_q \bar{\kappa}_k^{-1} \bar{V}_q \tag{25}$$

where (21) is used for $\alpha = a$ and (20) is used for $\alpha \in \{b, c, d, 1/\kappa\}$. These operators can be directly used on the right-hand sides of (10)–(13). Substituting (22)–(25) into (10)–(13) yields

$$\begin{aligned} \partial_t \bar{H}_k = &-\bar{P}_q \bar{a}_k^{-1} \bar{V}_q \Big[\bar{P}_q \left(\bar{b}_k \bar{V}_q \bar{H}_k + \bar{c}_k \bar{V}_q \bar{P}_k^H \right) \\ &+ \mu_k^{-1} \bar{\mathbb{C}}_k (\bar{E}_k, \bar{E}_{k'}, \bar{H}_k, \bar{H}_{k'}) \Big] \end{aligned} \tag{26}$$

$$\begin{aligned} \partial_t \bar{E}_k = &-\bar{P}_q \bar{a}_k^{-1} \bar{V}_q \Big[\bar{P}_q \left(\bar{b}_k \bar{V}_q \bar{E}_k + \bar{c}_k \bar{V}_q \bar{P}_k^E \right) \\ &- \epsilon_k^{-1} \bar{\mathbb{C}}_k (\bar{H}_k, \bar{H}_{k'}, \bar{E}_k, \bar{E}_{k'}) \Big] \end{aligned} \tag{27}$$

$$\partial_t \bar{P}_k^H = \bar{P}_q \left(\bar{\kappa}_k^{-1} \bar{V}_q \bar{H}_k - \bar{d}_k \bar{V}_q \bar{P}_k^H \right) \tag{28}$$

$$\partial_t \bar{P}_k^E = \bar{P}_q \left(\bar{\kappa}_k^{-1} \bar{V}_q \bar{E}_k - \bar{d}_k \bar{V}_q \bar{P}_k^E \right). \tag{29}$$

Equations (26)–(29) can be implemented in a matrix-free manner just like it is done in classical DG implementations [8]–[10], [28]–[31].

B. Computational Complexity

In DGTD with explicit time marching, all operations are localized within the elements. The memory required to store the mass matrices in the direct implementation of (10)–(13) scales with $K_{\text{PML}} \times 15\ N_p^2$, where 15 comes from the number of unknown components times the number of different mass matrices associated with different coefficients and K_{PML} is the number of elements in the PML. In the WAA formulation (26)–(29), the memory requirement reduces to $(K_{\text{PML}} \times 15\ N_q) + 2N_p N_q$, where $15\ N_q$ comes from the number of unknown components times the number of coefficient samples at the quadrature points and $2\ N_p N_q$ comes from $\bar{V}_q$ and $\bar{P}_q$ defined on the reference element. For simplified quadrature rules that are exact for up to polynomials of degree $2p+1$, $N_q \sim N_p$ [32], [33].

To compare the number of arithmetic operations required by the two implementations, one should first note that the curl operator $\bar{\mathbb{C}}$ is the same in both formulations. Computation of $\bar{\mathbb{C}}$ requires those of the spatial derivatives and the numerical flux [34]–[36]. Here, the memory access time is much more significant than the time required to carry out these computations because data from neighboring elements, which are discontinuous in memory, is required. Therefore, only the times required to complete the arithmetic operations of the remaining terms are compared. For the same reason, in practice, the time required to compute $\bar{\mathbb{C}}$ dominates the overall time required by the time marching, and the difference in the numbers of arithmetic operations as estimated below for the remaining terms is less significant (see the example in Section III).

In (10), the three matrix–vector multiplications and two vector–vector additions require $3\ N_p^2$ multiplication operations and $2\ N_p$ addition operations, respectively. In (26), the multiplication of $\bar{V}_q$ with a vector of length N_p and the multiplication of $\bar{P}_q$ with a vector of length N_q require $N_q N_p$ multiplication operations. The multiplication of a diagonal matrix with a vector (such as $\bar{b}_k \bar{v}$) requires N_q multiplication operations. As a result, excluding the computation of $\bar{\mathbb{C}}$, (26) requires $5\ N_q N_p + 3N_q$ multiplications and $N_q + N_p$ additions. For the auxiliary variable, the cost of (12) is $3\ N_p^2$ multiplications and N_p subtractions, while (28) requires $3\ N_q N_p + 2N_q$ multiplications and N_q subtractions. One can see that the number of operations in the WAA implementation is slightly higher than that in the direct implementation. However, as mentioned earlier, the time required by these operations is smaller than the time required to compute $\bar{\mathbb{C}}$, and therefore, overall times required by the two implementations are not that different.

III. Numerical Examples

In this section, the accuracy and the efficiency of the proposed WAA formulation are compared to those of the traditional PML implementations using numerical examples. To this end, five PML configurations/implementations are considered in these examples: 1) σ_u and/or κ_u, $u \in \{x, y, z\}$, are assumed constant inside the elements

on a paved mesh (EC-paved) [see Fig. 1(a)]; 2) σ_u and/or κ_u, $u \in \{x, y, z\}$, are assumed constant inside the elements on a layered mesh (EC-layered) [see Fig. 1(b)]; 3) σ_u and/or κ_u, $u \in \{x, y, z\}$, are allowed to vary inside the elements on a paved mesh (SV-paved) [see Fig. 1(c)]; 4) the same configuration in 3) but implemented using the proposed method with the WAA (SV-WAA-paved) [see Fig. 1(c)]; and 5) σ_u and/or κ_u, $u \in \{x, y, z\}$, are assumed constant inside the whole PML (CC). In all configurations/implementations, the order of the Lagrange polynomials $p \in \{1, 2, 3, 4, 5, \}$, which results in $N_p \in \{4, 10, 20, 35, 56\}$.

For configuration 1), the constant values in a given element are obtained by sampling σ_u and κ_u, $u \in \{x, y, z\}$, at that element's node that is farthest away from the PML interface (along the $\pm u$-direction). For configuration 2), to ensure that the element surfaces are strictly parallel to the axes, the PML mesh is built layer by layer and constant values in a given layer are obtained by sampling σ_u and κ_u, $u \in \{x, y, z\}$, at the outermost surface of that layer (along the $\pm u$-direction). For the WAA in implementation 4), the order of the Gaussian quadrature rule is $2p$, resulting in $N_q \in \{4, 11, 23, 44, 74\}$ [33].

In all examples, the background medium is free space and the excitation is a plane wave with electric field $\mathbf{E}(z, t) = E_0 \hat{\mathbf{x}} G(t - z/c_0)$, where $E_0 = 1$ V/m, c_0 is the speed of light in free space, and $G(t) = e^{(t-t_0)^2/4\tau^2}$ is a baseband Gaussian pulse with $\tau = 66.67$ ps and $t_0 = 15\tau$. The average edge lengths of all meshes used under this excitation are 0.4 cm.

First, a plane wave that is normally incident on the PML is considered. The computation domain is a rectangular box with dimensions 1.2 cm × 1.2 cm × 60 cm. Perfect electric conductor (PEC) and periodic boundary conditions are used on the outer boundary of the PML that is located perpendicular to the z direction and on the computation domain boundaries perpendicular to the x- and y-directions, respectively. The plane wave excitation is introduced on surface $z = 0$ and propagates in the $+z$-direction. The conductivity profile is described by $\sigma_z(z) = \sigma_{\max}[(z - z_0)/L_z]^{p_\sigma}$, where z_0 is the z-coordinate on the interface between PML and the computation domain, L_z is the thickness of the PML, and p_σ is the order of the profile. Note that $\sigma_z(z)$ is nonzero only when $|z| > |z_0|$. The values of these parameters are $z_0 = \pm 30$ cm, $L_z = 1.6$ cm, and also $\kappa_z(z) = 1$ both inside the PML and the computation domain.

In this example, five configurations/implementations are considered: EC-paved, EC-layered, SV-paved, SV-WAA-paved ($p_\sigma = 2$ for all four), and CC ($p_\sigma = 0$, yielding a constant conductivity profile in the whole PML). Their performances are compared for $p \in \{2, 3, 4, 5\}$. For all five groups of simulations, $\sigma_{\max}$ is scanned to find the minimum reflection that can be obtained for each case. Here, "reflection" is defined as the peak value of the absolute difference between the field computed at point $(0, 0, 0.4\,\text{cm})$ and the corresponding reference field computed at the same point. The reference field is computed under the same excitation, but the length of the computation domain in the z-direction is extended to 600 cm. This extension ensures that the reference field is free from PML reflections within the simulation time. To ensure that the discretization errors are at the same level, the meshes in the overlapped region between the computation domains of the test configuration and the extended computation domain are kept exactly the same, the average edge lengths of the mesh in the extended region are kept the same as those in the computation domain of the test configuration, and the solutions are obtained using the same value of p.

Fig. 2 shows that with increasing $\sigma_{\max}$, the reflection first decreases exponentially and then increases gradually. This is observed for all configurations/implementations and all values of p. When $\sigma_{\max}$ is small, the overall reflection is dominated by the reflection from the PEC boundary simply because the absorption inside the PML is not high enough. Therefore, in this regime, increasing $\sigma_{\max}$ elevates the absorption and reduces the amplitude of the wave reflected back into the computation domain exponentially. The numerical reflection (which is smaller than the reflection from the PEC boundary for small $\sigma_{\max}$) increases with increasing $\sigma_{\max}$ [11] and starts dominating the overall reflection as demonstrated in the figure by the gradual increase after the minimum point.

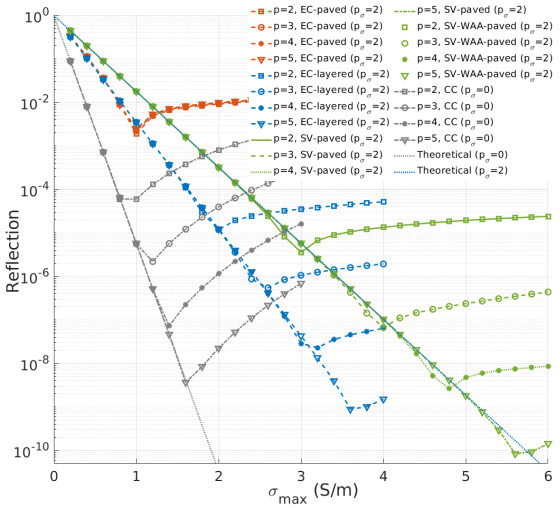

Fig. 2. Peak value of the reflected field's amplitude versus $\sigma_{\max}$ for different PML configurations/implementations and orders of Lagrange polynomials (p). The plane wave is normally incident on the PML.

For the EC-paved configuration, the reflection stays at a high level and does not decrease with increasing p. This is because of the large reflections from unoriented internal element surfaces. For the EC-layered, SV-layered, and SV-WAA-paved configurations, high-order convergence is observed, i.e., the reflection keeps on decreasing exponentially with increasing p. Still, the reflection for the SV-paved and SV-WAA-paved configurations is about 15 dB smaller than that for the EC-layered configuration. Note that this higher accuracy comes with the ease of meshing since a layered mesh (and conductivity profile) is not needed. Fig. 2 also shows that the SV-WAA-paved implementation performs exactly the same as the SV-paved direct implementation, which verifies the accuracy of the proposed method. Another conclusion from Fig. 2 is that the smoothly varying profile improves the performance of the PML. Fig. 2 also plots the theoretical reflection from the PML in the continuous space (i.e., without discretization) for $p_\sigma = 0$ and $p_\sigma = 2$ [1]. The slope of the $p_\sigma = 0$ curve is the same as that of the CC configuration, and the slope of the $p_\sigma = 2$ curve is the same as that of SV- and SV-WAA-paved configurations. For the EC-layered configuration, the effective conductivity is higher than that of the smoothly varying profile. Therefore, the slope of the corresponding reflection curve is larger than that of the theoretical one.

Next, scattering from a PEC sphere of radius 1 cm is considered. The computation domains and the PMLs for the EC-layered and SV- and SV-WAA-paved configurations are shown in Fig. 3(a) and (b), respectively. The plane wave excitation is introduced on the total-field scattered-field (TFSF) surface [shown in green in Fig. 3(a) and (b)]. The conductivity function is $\sigma_u(u) = \sigma_{\max}[(u - u_0)/L_u]^{p_\sigma}$, $u \in \{x, y, z\}$, $u_0 \in \{x_0, y_0, z_0\}$, where u_0 is the u-coordinate on the interface between PML and the computation domain, L_u is the thickness of the PML along the $\pm u$ direction, and p_σ is the order of the profile. Note that $\sigma_u(u)$ is nonzero only when $|u| > |u_0|$. The values of these parameters are $x_0 = y_0 = z_0 = \pm 2.2$ cm and $L_x = L_y = L_z = 1.2$ cm.

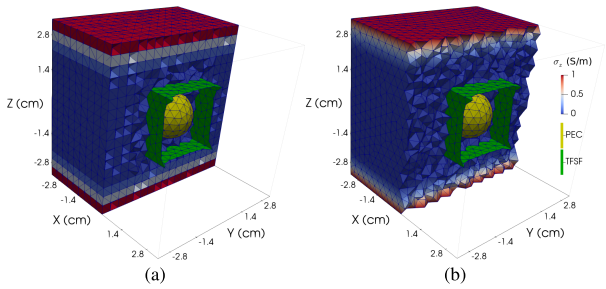

Fig. 3. Computation domains, meshes, and PML conductivity profiles (represented by color) used for (a) EC-layered and (b) SV-paved and SV-WAA-paved configurations.

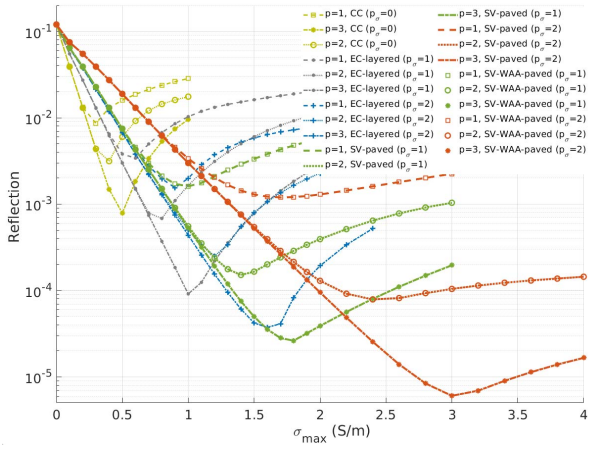

Fig. 4. Reflection of the scattered field versus $\sigma_{\max}$ for different PML configurations/implementations, orders of the Lagrange polynomials (p), and orders of the PML profile (p_σ). The scatterer is a PEC sphere of radius 1 cm.

Because the distance between the sphere surface and the PML is short, possibly evanescent scattered waves enter the PML with high grazing angles. A varying κ_u, $u \in \{x, y, z\}$, profile is employed to help with the absorption of these evanescent waves [13], [14]: $\kappa_u(u) = 1 + (\kappa_{\max} - 1)[(u - u_0)/L_u]^{p_\sigma}$ with $\kappa_{\max} = 2$. Note that inside the computation domain, $\kappa_u(u) = 1$. In this example, using the PEC or the first-order absorbing boundary condition [16] on the outer boundary of the PML gives similar results. The results presented here are obtained with the PEC boundary condition.

Four configurations/implementations are considered here: EC-layered, SV-paved, and SV-WAA-paved ($p_\sigma \in \{1, 2\}$ for all three) and CC ($p_\sigma = 0$ yielding a constant conductivity profile.) For the EC-layered configuration, the thickness of each PML mesh layer is 0.4 cm [see Fig. 3(a)]. Note that, for this example, generation of these layers is rather tedious since, in the corner region, one has to align all layer/element surfaces in all three directions. In contrast, for the SV- and SV-WAA-paved configurations [the same mesh is used—Fig. 3(b)], σ_u and κ_u values are simply obtained by sampling the corresponding profile functions at the nodes of the elements. This significantly simplifies the setups of the computation domain and the PML since even an explicit interface between the computation domain and the PML is not required [see Fig. 3(b)]. The performances of the three configurations are compared for $p \in \{1, 2, 3\}$.

$\sigma_{\max}$ is scanned to find the minimum reflection that could be reached for each case. Note that in this example, "reflection" is defined as the peak value of the absolute difference between the field computed at a probe point and the corresponding reference field computed at the same point. Ten different probe points (placed either in the TF or in the SF region) have been tested, and the results are consistent for all of them. The results below correspond to the probe point at (1.0 cm, 1.0 cm, 1.0 cm). The reference field is computed under the same excitation, but the distance between the sphere surface and the PML is extended to 12 cm. To ensure that the discretization errors are at the same level, the mesh in the overlapped region between the computation domain of the test configuration and the extended computation domain is kept exactly the same, the average edge lengths of the mesh in the extended region are kept the same as those in the computation domain of the test configuration, and the solutions are obtained using the same value of p.

TABLE I

COMPUTATIONAL COSTS OF THE SV- AND SV-WAA-PAVED IMPLEMENTATIONS*

p	N_p	N_q	memory (GB)		CPU time per step (s)	
			SV-paved	SV-WAA-paved	SV-paved	SV-WAA-paved
1	4	4	0.36	0.26	1.65	2.34
2	10	11	1.22	0.48	4.08	5.96
3	20	23	3.94	0.85	9.61	15.7
4	35	44	11.1	1.44	19.7	30.2
5	56	74	27.1	2.19	78.6	105

* Tested on a workstation with Intel Xeon(R) E5-2680 v4 CPU and 128GB memory. A single process is used. $K = 72,762$ and $K_{\mathrm{PML}} = 52,657$.

Fig. 4 plots the reflection for the three cases with different values of p and p_σ versus $\sigma_{\max}$. Clearly, the SV- and SV-WAA-paved configurations perform better than the CC configuration and the EC-layered configurations for every value of p_σ. The best performance is obtained with $p_\sigma = 2$. Note that further increasing p_σ degrades the PML performance for all configurations/implementations and all values of p since high conductivity values only appear at the very end of the PML when p_σ is high. Fig. 4 also shows that the SV-WAA-paved implementation performs exactly the same as the SV-paved direct implementation, which means that the error caused by the WAA of the mass matrices is below the level of the discretization error.

Table I compares the computational cost of the SV- and SV-WAA-paved implementations. The number of elements in the computation domain and the PML is 72 762 and 52 657, respectively. The memory requirements and the computation times presented in Table I correspond to those of the whole simulation (the computation domain and the PML combined). The savings that come with the use of the WAA depend on the ratio of the numbers of elements in these two regions. The table shows that with increasing p, the memory requirement increases dramatically for the SV-paved direct implementation but only modestly for the SV-WAA implementation. For $p = 5$, the memory requirement of the SV-paved implementation is 12.4 times that of the SV-WAA-paved implementation. The computation time required by the SV-WAA-paved implementation per time step is slightly larger than that required by the SV-paved implementation due to the increased number of arithmetic operations (see Section II-B).

It should also be noted here that, in practice, a DGTD algorithm is usually parallelized. The difference in times required for updating different elements is relatively small and can be easily compensated by allocating a smaller number of elements for those MPI processes containing PML elements. In the numerical results presented here, assigning a weight of 2 for PML elements in ParMetis [37], [38] yields a good load balance. The efficiency of the DGTD scheme can further be increased by combining the WAA with techniques making use of nonconformal meshes [39] and/or p-refinement for the PML.

IV. CONCLUSION

A PML implementation that allows the attenuation coefficient to vary inside the discretization elements yields a smaller numerical reflection from the interface between the PML and the computation

domain and significantly simplifies the meshing process. However, these advantages come at the cost of increased memory footprint since a different mass matrix has to be stored for every discretization element. In this work, this memory requirement is reduced by applying WAA to the mass matrices without abandoning the advantages listed earlier. Indeed, numerical results demonstrate that the PML with smoothly increasing conductivity profile as implemented with the proposed method performs better than the PML implemented using elementwise constant conductivity profile and that the higher order accuracy of the solution is maintained.

The proposed method is especially useful for simulations running on shared-memory systems where the high memory requirement of smoothly varying PMLs could be a bottleneck. For simulations running on distributed-memory systems, the memory requirement of a single computing node is also reduced and a better load balance could be reached with a slightly adjusted weight in the domain partition.

ACKNOWLEDGMENT

The authors would like to thank the KAUST Supercomputing Laboratory (KSL) for providing the required computational resources.

REFERENCES

[1] J.-P. Berenger, "A perfectly matched layer for the absorption of electromagnetic waves," *J. Comput. Phys.*, vol. 114, no. 2, pp. 185–200, Oct. 1994.

[2] W. C. Chew and W. H. Weedon, "A 3D perfectly matched medium from modified Maxwell's equations with stretched coordinates," *Microw. Opt. Technol. Lett.*, vol. 7, no. 13, pp. 599–604, Sep. 1994.

[3] A. Taflove and S. C. Hagness, *Computational Electrodynamics: The Finite-difference Time-Domain Method*. Norwood, MA, USA: Artech House, 2005.

[4] J.-M. Jin, *The Finite Element Method in Electromagnetics*. Hoboken, NJ, USA: Wiley, 2015.

[5] J. S. Hesthaven and T. Warburton, "Nodal high-order methods on unstructured grids: I. Time-domain solution of Maxwell's equations," *J. Comput. Phys.*, vol. 181, no. 1, pp. 186–221, 2002.

[6] B. Cockburn, F. Li, and C.-W. Shu, "Locally divergence-free discontinuous Galerkin methods for the Maxwell equations," *J. Comput. Phys.*, vol. 194, no. 2, pp. 588–610, Mar. 2004.

[7] T. Lu, P. Zhang, and W. Cai, "Discontinuous Galerkin methods for dispersive and lossy Maxwell's equations and PML boundary conditions," *J. Comput. Phys.*, vol. 200, no. 2, pp. 549–580, Nov. 2004.

[8] J. Hesthaven and T. Warburton, *Nodal Discontinuous Galerkin Methods: Algorithms, Analysis, and Applications*. New York, NY, USA: Springer, 2008.

[9] S. D. Gedney *et al.*, "The discontinuous Galerkin finite-element time-domain method solution of Maxwell's equations," *Appl. Comput. Electromagn. Soc. J.*, vol. 24, no. 2, p. 129, 2009.

[10] G. C. Cohen and S. Pernet, *Finite Element Discontinuous Galerkin Methods for Transient Wave Equations*. Berlin, Germany: Springer, 2017.

[11] W. C. Chew and J. M. Jin, "Perfectly matched layers in the discretized space: An analysis and optimization," *Electromagnetics*, vol. 16, no. 4, pp. 325–340, Jul. 1996.

[12] R. Zhang *et al.*, "Optimization of the periodic PML for SEM," *IEEE Trans. Electromagn. Compat.*, vol. 61, no. 5, pp. 1578–1585, Oct. 2019.

[13] J.-P. Bérenger, "Perfectly matched layer (PML) for computational electromagnetics," *Synth. Lectures Comput. Electromagn.*, vol. 2, no. 1, pp. 1–117, Jan. 2007.

[14] S. D. Gedney, "Introduction to the finite-difference time-domain (FDTD) method for electromagnetics," *Synth. Lectures Comput. Electromagn.*, vol. 6, no. 1, pp. 1–250, Jan. 2011.

[15] G. Chen, L. Zhao, W. Yu, S. Yan, K. Zhang, and J.-M. Jin, "A general scheme for the discontinuous Galerkin time-domain modeling and S-Parameter extraction of inhomogeneous waveports," *IEEE Trans. Microw. Theory Techn.*, vol. 66, no. 4, pp. 1701–1712, Apr. 2018.

[16] L. D. Angulo, J. Alvarez, M. F. Pantoja, S. G. Garcia, and A. R. Bretones, "Discontinuous Galerkin time domain methods in computational electrodynamics: State of the art," *Forum Electromagn. Res. Methods Appl. Technol.*, vol. 10, pp. 1–24, Aug. 2015.

[17] L. M. D. Angulo, "Time domain discontinuous Galerkin methods for Maxwell equations," Ph.D. dissertation, Dept. Phys., Univ. Granada, Granada, Spain, 2014.

[18] K. Sankaran, "Accurate domain truncation techniques for time-domain conformal methods," Ph.D. dissertation, Dept. Inf. Technol. Elect. Eng., ETH Zürich, Zürich, Switzerland, 2007.

[19] J. Niegemann, M. König, K. Stannigel, and K. Busch, "Higher-order time-domain methods for the analysis of nano-photonic systems," *Photon. Nanostruct. Fundam. Appl.*, vol. 7, no. 1, pp. 2–11, Feb. 2009.

[20] J. Chan, R. J. Hewett, and T. Warburton, "Weight-adjusted discontinuous Galerkin methods: Wave propagation in heterogeneous media," *SIAM J. Sci. Comput.*, vol. 39, no. 6, pp. A2935–A2961, Jan. 2017.

[21] K. Guo and J. Chan, "Bernstein-Bézier weight-adjusted discontinuous Galerkin methods for wave propagation in heterogeneous media," *J. Comput. Phys.*, vol. 400, Jan. 2020, Art. no. 108971.

[22] K. Shukla, J. Chan, M. V. de Hoop, and P. Jaiswal, "A weight-adjusted discontinuous Galerkin method for the poroelastic wave equation: Penalty fluxes and micro-heterogeneities," *J. Comput. Phys.*, vol. 403, Feb. 2020, Art. no. 109061.

[23] L. Chen and H. Bagci, "Steady-state simulation of semiconductor devices using discontinuous Galerkin methods," *IEEE Access*, vol. 8, pp. 16203–16215, 2020.

[24] L. Chen and H. Bagci, "Multiphysics simulation of plasmonic photoconductive devices using discontinuous Galerkin methods," *IEEE J. Multiscale Multiphys. Comput. Techn.*, vol. 5, pp. 188–200, 2020.

[25] P. Li, Y. Shi, L. J. Jiang, and H. Bagci, "DGTD analysis of electromagnetic scattering from penetrable conductive objects with IBC," *IEEE Trans. Antennas Propag.*, vol. 63, no. 12, pp. 5686–5697, Dec. 2015.

[26] P. Li, L. J. Jiang, and H. Bagci, "Transient analysis of dispersive power-ground plate pairs with arbitrarily shaped antipads by the DGTD method with wave port excitation," *IEEE Trans. Electromagn. Compat.*, vol. 59, no. 1, pp. 172–183, Feb. 2017.

[27] P. Li, L. J. Jiang, and H. Bagci, "Discontinuous Galerkin time-domain modeling of graphene nanoribbon incorporating the spatial dispersion effects," *IEEE Trans. Antennas Propag.*, vol. 66, no. 7, pp. 3590–3598, Jul. 2018.

[28] M. Liu, K. Sirenko, and H. Bagci, "An efficient discontinuous Galerkin finite element method for highly accurate solution of Maxwell equations," *IEEE Trans. Antennas Propag.*, vol. 60, no. 8, pp. 3992–3998, Aug. 2012.

[29] L. Chen and H. Bagci, "A discontinuous Galerkin framework for multiphysics simulation of photoconductive devices," in *Proc. Int. Appl. Comput. Electromagn. Soc. Symp.*, Apr. 2019, pp. 1–2.

[30] K. Sirenko, M. Liu, and H. Bagci, "Incorporation of exact boundary conditions into a discontinuous Galerkin finite element method for accurately solving 2D time-dependent Maxwell equations," *IEEE Trans. Antennas Propag.*, vol. 61, no. 1, pp. 472–477, Jan. 2013.

[31] L. Chen and H. Bagci, "A unit-cell discontinuous Galerkin scheme for analyzing plasmonic photomixers," in *Proc. IEEE Int. Symp. Antennas Propag.*, Jul. 2019, pp. 1069–1070.

[32] R. Cools, "Monomial cubature rules since, 'Strou': A compilation—Part 2," *J. Comput. Appl. Math.*, vol. 112, nos. 1–2, pp. 21–27, Nov. 1999.

[33] H. Xiao and Z. Gimbutas, "A numerical algorithm for the construction of efficient quadrature rules in two and higher dimensions," *Comput. Math. Appl.*, vol. 59, no. 2, pp. 663–676, Jan. 2010.

[34] L. Chen, M. Dong, and H. Bagci, "Modeling floating potential conductors using discontinuous Galerkin method," *IEEE Access*, vol. 8, pp. 7531–7538, 2020.

[35] K. Sirenko, Y. Sirenko, and H. Bağci, "Exact absorbing boundary conditions for periodic three-dimensional structures: Derivation and implementation in discontinuous Galerkin time-domain method," *IEEE J. Multiscale Multiphys. Comput. Techn.*, vol. 3, pp. 108–120, 2018.

[36] L. Chen, M. Dong, P. Li, and H. Bagci, "A hybridizable discontinuous Galerkin method for simulation of electrostatic problems with floating potential conductors," in *Proc. Int. J. Numer. Modelling, Electron. Netw., Devices Fields*, Sep. 2020, p. e2894.

[37] G. Karypis and V. Kumar, "A parallel algorithm for multilevel graph partitioning and sparse matrix ordering," *J. Parallel Distrib. Comput.*, vol. 48, no. 1, pp. 71–95, Jan. 1998.

[38] L. Chen and H. Bagci, "An MPI-based parallel multiphysics discontinuous Galerkin framework for photoconductive devices," in *Proc. Prog. Electromagn. Res. Symp.*, Xiamen, China, Dec. 2019, pp. 10–12.

[39] Q. Sun, R. Zhang, Q. Zhan, and Q. H. Liu, "A novel coupling algorithm for perfectly matched layer with wave equation-based discontinuous Galerkin time-domain method," *IEEE Trans. Antennas Propag.*, vol. 66, no. 1, pp. 255–261, Jan. 2018.