FAME: Fast Algorithms for Maxwell's Equations for Three-Dimensional Photonic Crystals

TSUNG-MING HUANG, National Taiwan Normal University
TIEXIANG LI*, Southeast University and Nanjing Center for Applied Mathematics
JIA-WEI LIN, National Chiao Tung University
WEN-WEI LIN, National Chiao Tung University
XING-LONG LYU, Southeast University
SHENG WANG, National Chiao Tung University

In this article, we propose the Fast Algorithms for Maxwell's Equations (FAME) package for solving Maxwell's equations for modeling three-dimensional photonic crystals. FAME combines the null-space free method with fast Fourier transform (FFT)-based matrix-vector multiplications to solve the generalized eigenvalue problems (GEPs) arising from Yee's discretization. The GEPs are transformed into a null-space free standard eigenvalue problem with a Hermitian positive-definite coefficient matrix. The computation times for FFT-based matrix-vector multiplications with matrices of dimension 7 million are only 0.33 and 3.6×10^{-3} seconds using MATLAB with an Intel Xeon CPU and CUDA C++ programming with a single NVIDIA Tesla P100 GPU, respectively. Such multiplications significantly reduce the computational costs of the conjugate gradient method for solving linear systems. We successfully use FAME on a single P100 GPU to solve a set of GEPs with matrices of dimension more than 19 million, in 127 to 191 seconds per problem. These results demonstrate the potential of our proposed package to enable large-scale numerical simulations for novel physical discoveries and engineering applications of photonic crystals.

ACM Reference Format:
Tsung-Ming Huang, Tiexiang Li, Jia-Wei Lin, Wen-Wei Lin, Xing-Long Lyu, and Sheng Wang. 2020. FAME: Fast Algorithms for Maxwell's Equations for Three-Dimensional Photonic Crystals. 1, 1 (December 2020), 26 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

*Corresponding author

Authors' addresses: Tsung-Ming Huang, min@ntnu.edu.tw, Department of Mathematics, National Taiwan Normal University, Taipei 116, Taiwan; Tiexiang Li, txli@seu.edu.cn, School of Mathematics, Southeast University, Nanjing 211189, China, Nanjing Center for Applied Mathematics, Nanjing 211135, China; Jia-Wei Lin, jiawei.am05g@g2.nctu.edu.tw, Department of Applied Mathematics, National Chiao Tung University, Hsinchu 300, Taiwan; Wen-Wei Lin, wwlin@math.nctu.edu.tw, Department of Applied Mathematics, National Chiao Tung University, Hsinchu 300, Taiwan; Xing-Long Lyu, lxl_math@seu.edu.cn, School of Mathematics, Southeast University, Nanjing 211189, China; Sheng Wang, shengwang.am08g@nctu.edu.tw, Department of Applied Mathematics, National Chiao Tung University, Hsinchu 300, Taiwan.

1 INTRODUCTION

The propagation of electromagnetic waves in bi-isotropic complex media can be mathematically modeled using the three-dimensional (3D) Maxwell's equations:

$$\nabla \times E = \imath\omega B, \quad \nabla \times H = -\imath\omega D, \tag{1a}$$

$$\nabla \cdot B = 0, \qquad \nabla \cdot D = 0, \tag{1b}$$

where ω is the frequency; E and H are the electric and magnetic fields, respectively; and B and D are the magnetic and electric flux densities, respectively, which satisfy the constitutive relations

$$B = \mu H + \zeta E \quad \text{and} \quad D = \varepsilon E + \xi H, \tag{2}$$

where μ and ε are the magnetic permeability and electric permittivity, respectively, and $\zeta = \xi^*$ are magnetoelectric parameters. According to the well-known Bloch theory [21], the E and H fields in (1) on a prescribed crystal lattice $\{\tilde{\mathbf{a}}_\ell\}_{\ell=1}^3$ are required to satisfy the following quasiperiodic condition [32]:

$$E(\mathbf{x} + \tilde{\mathbf{a}}_\ell) = \mathbf{e}^{\imath 2\pi \mathbf{k}\cdot\tilde{\mathbf{a}}_\ell} E(\mathbf{x}), \quad H(\mathbf{x} + \tilde{\mathbf{a}}_\ell) = \mathbf{e}^{\imath 2\pi \mathbf{k}\cdot\tilde{\mathbf{a}}_\ell} H(\mathbf{x}), \ \ell = 1, 2, 3, \tag{3}$$

where $2\pi\mathbf{k}$ is the Bloch wave vector within the first Brillouin zone [19] and the $\tilde{\mathbf{a}}_\ell$ are the lattice translation vectors.

In this paper, we consider the case of dielectric photonic crystals (PCs) with $\mu \equiv 1$, $\varepsilon > 0$ and $\xi = \zeta = 0$ in (1), where ε is a material-dependent piecewise constant function. The associated governing equation can be simplified as

$$\nabla \times \mu^{-1} \nabla \times E = \omega^2 \varepsilon E, \quad \nabla \cdot (\varepsilon E) = 0. \tag{4}$$

Additionally, in a dispersive metallic material [7–9, 25, 27, 34], ε in (4) is dependent on the frequency ω, that is, $\varepsilon = \varepsilon(\mathbf{x}, \omega)$ at the position $\mathbf{x} \in \mathbb{R}^3$. Such materials can exhibit negative permittivities at certain frequencies, such as below the plasma frequency. On the other hand, left-handed materials or negative-index materials have $\mu < 0$ and $\varepsilon < 0$. Although no such material exists in nature, artificial metamaterials with periodic structures have been proposed in [30, 31] that exhibit the characteristic properties of such materials, including a negative refractive index. The associated B and D satisfy the constitutive relations given in (2).

A number of methods are available for discretizing (1), including the plane-wave expansion method [20], the finite-difference frequency-domain method (FDFD) [3, 10, 38–40], the finite element method [1, 2, 5, 6, 18, 28, 29], and the mixed finite element method [22, 23], to name a few. Due to the divergence-free condition given in (1b), the dimensions of the null space of the resulting discrete generalized eigenvalue problem (GEP) account for one-third of the total dimensions. The presence of such a large null space will seriously affect the convergence to the desired solution to the GEP and thus presents a numerical challenge. Another challenge is to solve the corresponding linear system, which is extremely large in size, in each step of an iterative eigensolver. For the case of isotropic media, Yee's finite-difference scheme [39] is used to discretize (1), which results in a GEP. It has been shown that when using Yee's scheme, these challenges can be resolved by applying the null-space free techniques presented in [11, 12, 14]. With the help of this null-space free method, the GEPs resulting from (1) and (4) are transformed into a null-space free GEP (NFGEP) and a null-space free standard eigenvalue problem (NFSEP), respectively. Moreover, when solving

this NFSEP or NFGEP, the necessary matrix-vector multiplications can be significantly accelerated by means of the fast Fourier transform (FFT).

The efficiency of the null-space free method combined with FFT-based matrix-vector multiplications has been demonstrated in [11, 12, 16]. In this paper, we will focus on 3D nondispersive PCs as described in (4). The main goals of this paper are threefold: (i) to integrate all of the techniques discussed above, including the null-space free method and FFT-based matrix-vector multiplications, into a package called Fast Algorithms for Maxwell's Equations (FAME) for the computation of the band structures of 3D PCs with all Bravais lattices; (ii) to implement FAME on two architectures, namely, CUDA with a GPU accelerator and MATLAB without a GPU architecture; and (iii) to provide various material structures [11, 24, 26, 35, 37] in FAME as benchmark PCs, which users can easily use to efficiently simulate the associated band structures. Furthermore, users can also define material structures themselves to be used in FAME.

This paper is organized as follows. In Section 2, we introduce 7 lattice systems and 14 Bravais lattices in 3D Euclidean space. The matrix representations of the discrete single curl operator and the associated singular value decomposition (SVD) are presented in Section 3. In Section 4, we introduce the FFT-based matrix-vector multiplications proposed in [16] and demonstrate their efficiency when implemented in the MATLAB environment and on a single NVIDIA GPU architecture. The application of the SVD as an efficient preconditioner for solving GEPs and the null-space free method for solving NFSEPs are introduced in Section 5. Numerical comparisons show that the null-space free method outperforms the SVD-based preconditioning method. In Section 6, we introduce the integration of all of the above schemes into the FAME package and present four benchmark PCs to demonstrate the timing performance of FAME. Finally, concluding remarks are given in Section 7.

Notations. For matrices A and B and vector $\mathbf{v}$, $A^\top$ and A^* are the transpose and conjugate transpose, respectively; $\mathrm{diag}(A, B)$ denotes a diagonal block matrix whose block entries are matrices A and B; $\mathrm{diag}(A)$ denotes a vector whose entries are the diagonal entries of the matrix A; $\mathrm{diag}(\mathbf{v})$ denotes a diagonal matrix whose diagonal entries are the entries of the vector $\mathbf{v}$; $A \otimes B$, $A \oplus B = \mathrm{diag}(A, B)$, and $A \odot B = [a_{ij} * b_{ij}]$ are the Kronecker product, the direct sum, and the Hadamard product of matrices A and B, respectively; $\mathrm{vec}(A)$ denotes a vector obtained by stacking the columns of the matrix A on top of one another.

2 LATTICE TRANSLATION VECTORS

A 3D PC is a periodic lattice composed of dielectric materials. In crystallography, the so-called primitive cell of a crystal is a fundamental domain under translational symmetry and contains just one lattice point. In fact, a 3D primitive cell, denoted by Ω_p, is a (slanted) parallelepiped formed by three lattice translation vectors, $\tilde{\mathbf{a}}_1$, $\tilde{\mathbf{a}}_2$ and $\tilde{\mathbf{a}}_3$, as illustrated in Figure 1(a). At present, millions of crystals are known, and each crystal has a different nature. As shown in [26], such crystals with different lattice translation vectors can be classified into 7 lattice systems and 14 Bravais lattices in 3D Euclidean space, which are listed below.

(1) Cubic system
  (a) Cubic lattice (CUB)

$$\tilde{\mathbf{a}}_1 = \begin{bmatrix} a & 0 & 0 \end{bmatrix}^\top, \quad \tilde{\mathbf{a}}_2 = \begin{bmatrix} 0 & a & 0 \end{bmatrix}^\top, \quad \tilde{\mathbf{a}}_3 = \begin{bmatrix} 0 & 0 & a \end{bmatrix}^\top;$$

(b) Body-centered Cubic lattice (BCC)

$$\tilde{\mathbf{a}}_1 = \begin{bmatrix} -\frac{a}{2} & \frac{a}{2} & \frac{a}{2} \end{bmatrix}^\top, \quad \tilde{\mathbf{a}}_2 = \begin{bmatrix} \frac{a}{2} & -\frac{a}{2} & \frac{a}{2} \end{bmatrix}^\top, \quad \tilde{\mathbf{a}}_3 = \begin{bmatrix} \frac{a}{2} & \frac{a}{2} & -\frac{a}{2} \end{bmatrix}^\top;$$

(c) Face-centered Cubic lattice (FCC)

$$\tilde{\mathbf{a}}_1 = \begin{bmatrix} 0 & \frac{a}{2} & \frac{a}{2} \end{bmatrix}^\top, \quad \tilde{\mathbf{a}}_2 = \begin{bmatrix} \frac{a}{2} & 0 & \frac{a}{2} \end{bmatrix}^\top, \quad \tilde{\mathbf{a}}_3 = \begin{bmatrix} \frac{a}{2} & \frac{a}{2} & 0 \end{bmatrix}^\top;$$

(2) Tetragonal system ($a \neq c$)

(a) Tetragonal lattice (TET)

$$\tilde{\mathbf{a}}_1 = \begin{bmatrix} a & 0 & 0 \end{bmatrix}^\top, \quad \tilde{\mathbf{a}}_2 = \begin{bmatrix} 0 & a & 0 \end{bmatrix}^\top, \quad \tilde{\mathbf{a}}_3 = \begin{bmatrix} 0 & 0 & c \end{bmatrix}^\top;$$

(b) Body-centered Tetragonal lattice (BCT)

$$\tilde{\mathbf{a}}_1 = \begin{bmatrix} -\frac{a}{2} & \frac{a}{2} & \frac{c}{2} \end{bmatrix}^\top, \quad \tilde{\mathbf{a}}_2 = \begin{bmatrix} \frac{a}{2} & -\frac{a}{2} & \frac{c}{2} \end{bmatrix}^\top, \quad \tilde{\mathbf{a}}_3 = \begin{bmatrix} \frac{a}{2} & \frac{a}{2} & -\frac{c}{2} \end{bmatrix}^\top;$$

(3) Orthorhombic system ($a < b < c$)

(a) Orthorhombic lattice (ORC)

$$\tilde{\mathbf{a}}_1 = \begin{bmatrix} a & 0 & 0 \end{bmatrix}^\top, \quad \tilde{\mathbf{a}}_2 = \begin{bmatrix} 0 & b & 0 \end{bmatrix}^\top, \quad \tilde{\mathbf{a}}_3 = \begin{bmatrix} 0 & 0 & c \end{bmatrix}^\top;$$

(b) Body-centered Orthorhombic lattice (ORCI)

$$\tilde{\mathbf{a}}_1 = \begin{bmatrix} -\frac{a}{2} & \frac{b}{2} & \frac{c}{2} \end{bmatrix}^\top, \quad \tilde{\mathbf{a}}_2 = \begin{bmatrix} \frac{a}{2} & -\frac{b}{2} & \frac{c}{2} \end{bmatrix}^\top, \quad \tilde{\mathbf{a}}_3 = \begin{bmatrix} \frac{a}{2} & \frac{b}{2} & -\frac{c}{2} \end{bmatrix}^\top;$$

(c) Face-centered Orthorhombic lattice (ORCF)

$$\tilde{\mathbf{a}}_1 = \begin{bmatrix} 0 & \frac{b}{2} & \frac{c}{2} \end{bmatrix}^\top, \quad \tilde{\mathbf{a}}_2 = \begin{bmatrix} \frac{a}{2} & 0 & \frac{c}{2} \end{bmatrix}^\top, \quad \tilde{\mathbf{a}}_3 = \begin{bmatrix} \frac{a}{2} & \frac{b}{2} & 0 \end{bmatrix}^\top;$$

(d) C-centered Orthorhombic lattice (ORCC)

$$\tilde{\mathbf{a}}_1 = \begin{bmatrix} \frac{a}{2} & -\frac{b}{2} & 0 \end{bmatrix}^\top, \quad \tilde{\mathbf{a}}_2 = \begin{bmatrix} \frac{a}{2} & \frac{b}{2} & 0 \end{bmatrix}^\top, \quad \tilde{\mathbf{a}}_3 = \begin{bmatrix} 0 & 0 & c \end{bmatrix}^\top;$$

(4) Hexagonal system ($a \neq c$)

(a) Hexagonal lattice (HEX)

$$\tilde{\mathbf{a}}_1 = \begin{bmatrix} \frac{a}{2} & -\frac{a\sqrt{3}}{2} & 0 \end{bmatrix}^\top, \quad \tilde{\mathbf{a}}_2 = \begin{bmatrix} \frac{a}{2} & \frac{a\sqrt{3}}{2} & 0 \end{bmatrix}^\top, \quad \tilde{\mathbf{a}}_3 = \begin{bmatrix} 0 & 0 & c \end{bmatrix}^\top;$$

(5) Monoclinic system ($a, b \leq c$; $\alpha \leq \frac{\pi}{2}$)

(a) Monoclinic lattice (MCL)

$$\tilde{\mathbf{a}}_1 = \begin{bmatrix} a & 0 & 0 \end{bmatrix}^\top, \quad \tilde{\mathbf{a}}_2 = \begin{bmatrix} 0 & b & 0 \end{bmatrix}^\top, \quad \tilde{\mathbf{a}}_3 = \begin{bmatrix} 0 & c\cos\alpha & c\sin\alpha \end{bmatrix}^\top;$$

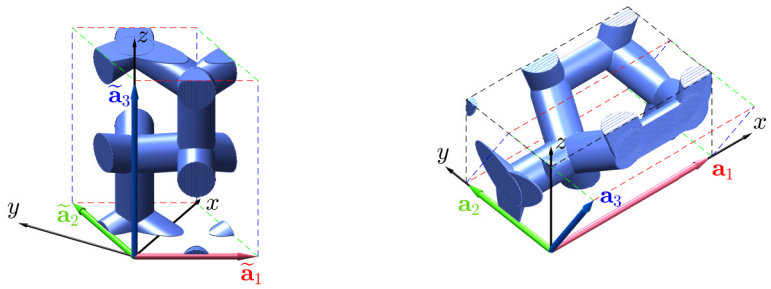
(b) C-centered Monoclinic lattice (MCLC)

$$\tilde{\mathbf{a}}_1 = \begin{bmatrix} \frac{a}{2} & \frac{b}{2} & 0 \end{bmatrix}^\top, \quad \tilde{\mathbf{a}}_2 = \begin{bmatrix} -\frac{a}{2} & \frac{b}{2} & 0 \end{bmatrix}^\top, \quad \tilde{\mathbf{a}}_3 = \begin{bmatrix} 0 & c\cos\alpha & c\sin\alpha \end{bmatrix}^\top;$$

(6) Rhombohedral system ($\alpha \neq \frac{\pi}{2}$)

(a) Rhombohedral lattice (RHL)

$$\tilde{\mathbf{a}}_1 = \begin{bmatrix} a\cos\frac{\alpha}{2} & -a\sin\frac{\alpha}{2} & 0 \end{bmatrix}^\top, \quad \tilde{\mathbf{a}}_2 = \begin{bmatrix} a\cos\frac{\alpha}{2} & a\sin\frac{\alpha}{2} & 0 \end{bmatrix}^\top, \quad \tilde{\mathbf{a}}_3 = \begin{bmatrix} \frac{a\cos\alpha}{\cos\frac{\alpha}{2}} & 0 & a\sqrt{1-\frac{\cos^2\alpha}{\cos^2\frac{\alpha}{2}}} \end{bmatrix}^\top;$$



(a) Primitive cell (b) Computational cell

Fig. 1. Illustration of the 3D primitive cell Ω_p and computational cell Ω_c of a hexagonal lattice.

(7) Triclinic system ($a \neq b \neq c$; $\alpha \neq \beta \neq \gamma$)

(a) Triclinic lattice (TRI)

$$\tilde{\mathbf{a}}_1 = \begin{bmatrix} a & 0 & 0 \end{bmatrix}^\top, \quad \tilde{\mathbf{a}}_2 = \begin{bmatrix} b\cos\gamma & b\sin\gamma & 0 \end{bmatrix}^\top,$$

$$\tilde{\mathbf{a}}_3 = \begin{bmatrix} c\cos\beta & \frac{c}{\sin\gamma}(\cos\alpha - \cos\beta\cos\gamma) & \frac{c}{\sin\gamma}\sqrt{\sin^2\gamma - \cos^2\alpha - \cos^2\beta + 2\cos\alpha\cos\beta\cos\gamma} \end{bmatrix}^\top;$$

Due to the misalignment between the lattice vectors $\{\tilde{\mathbf{a}}_\ell\}_{\ell=1}^3$ and the Cartesian coordinate, a finite-difference scheme used for the discretization of (4) in the slanted parallelepiped Ω_p (as shown in Figure 1(a)) may produce a large number of stair errors. Instead, we use the procedure described by (5) below to find new vectors $\mathbf{a}_1$, $\mathbf{a}_2$, and $\mathbf{a}_3$ to redefine an equivalent cuboid computational cell $\Omega_c = [0, \mathbf{a}_1(1)] \times [0, \mathbf{a}_2(2)] \times [0, \mathbf{a}_3(3)]$ which align with the Cartesian coordinates exactly, as shown in Figure 1(b). First, we compute the QR factorization with column pivoting of $[\tilde{\mathbf{a}}_1, \tilde{\mathbf{a}}_2, \tilde{\mathbf{a}}_3]$:

$$\widetilde{Q}^\top \begin{bmatrix} \tilde{\mathbf{a}}_1 & \tilde{\mathbf{a}}_2 & \tilde{\mathbf{a}}_3 \end{bmatrix} \Pi = R, \tag{5a}$$

where $\widetilde{Q}$ is orthonormal and R is upper triangular. Then, we take

$$\begin{bmatrix} \mathbf{a}_1 & \mathbf{a}_2 & \mathbf{a}_3 \end{bmatrix} \equiv SR, \quad Q = \widetilde{Q}S, \tag{5b}$$

where $S = \mathrm{diag}(\mathrm{sign}(R(1,1)), \mathrm{sign}(R(2,2)), \mathrm{sign}(R(3,3)))$. A 3D crystal can now be periodically generated along these new vectors $\mathbf{a}_1$, $\mathbf{a}_2$, and $\mathbf{a}_3$, and the quasiperiodic condition given in (3) can be rewritten as

$$E(\mathbf{x} + \mathbf{a}_\ell) = \mathbf{e}^{\imath 2\pi \mathbf{k}\cdot\mathbf{a}_\ell} E(\mathbf{x}), \quad H(\mathbf{x} + \mathbf{a}_\ell) = \mathbf{e}^{\imath 2\pi \mathbf{k}\cdot\mathbf{a}_\ell} H(\mathbf{x}), \ \ell = 1, 2, 3. \tag{6}$$

We describe the procedure for generating $\mathbf{a}_1$, $\mathbf{a}_2$, and $\mathbf{a}_3$ in Algorithm 1.

Algorithm 1 [14] Constructing computational lattice translation vectors

Input: Original lattice translation vectors $\tilde{\mathbf{a}}_1$, $\tilde{\mathbf{a}}_2$, and $\tilde{\mathbf{a}}_3$.
Output: Computational lattice vectors $\mathbf{a}_1$, $\mathbf{a}_2$, $\mathbf{a}_3$, a rotation matrix Q and a permutation matrix Π.
1: Compute the QR factorization with column pivoting of $\begin{bmatrix}\tilde{\mathbf{a}}_1 & \tilde{\mathbf{a}}_2 & \tilde{\mathbf{a}}_3\end{bmatrix}$ such that

$$\begin{bmatrix}\tilde{\mathbf{a}}_1 & \tilde{\mathbf{a}}_2 & \tilde{\mathbf{a}}_3\end{bmatrix}\Pi = \widetilde{Q}R$$

where Π is a permutation, $\widetilde{Q}$ is orthonormal, and R is upper triangular.
2: Set $S := \mathrm{diag}\{\mathrm{sign}(R(1,1)), \mathrm{sign}(R(2,2)), \mathrm{sign}(R(3,3))\}$ and take

$$\begin{bmatrix}\mathbf{a}_1 & \mathbf{a}_2 & \mathbf{a}_3\end{bmatrix} = SR; \quad Q = \widetilde{Q}S.$$

3 SINGULAR VALUE DECOMPOSITION

Let the positive integers n_1, n_2 and n_3 be the mesh sizes along the x-, y- and z-axes, respectively, $n = n_1 n_2 n_3$ be the grid number, and let $\delta_x = \mathbf{a}_1(1)/n_1$, $\delta_y = \mathbf{a}_2(2)/n_2$ and $\delta_z = \mathbf{a}_3(3)/n_3$ be the corresponding mesh lengths. We define three nonnegative quantities ρ_i $(i = 1, 2, 3)$ as follows:

$$\rho_1 = \begin{cases} 0, & \text{if } \mathbf{a}_2(1) \geq 0, \\ 1, & \text{if } \mathbf{a}_2(1) < 0; \end{cases} \quad \rho_2 = \begin{cases} 0, & \text{if } \mathbf{a}_3(1) \geq 0, \\ 1, & \text{if } \mathbf{a}_3(1) < 0; \end{cases} \quad \rho_3 = \begin{cases} 0, & \text{if } \mathbf{a}_3(2) \geq 0, \\ 1, & \text{if } \mathbf{a}_3(2) < 0. \end{cases} \tag{7}$$

We also define

$$m_1 = \mathrm{round}\left(\frac{\mathbf{a}_2(1) + \rho_1 \mathbf{a}_1(1)}{\delta_x}\right) \leq n_1, \ m_2 = \mathrm{round}\left(\frac{\mathbf{a}_3(1) + \rho_2 \mathbf{a}_1(1)}{\delta_x}\right) < n_1, \ m_3 = \mathrm{round}\left(\frac{\mathbf{a}_3(2) + \rho_3 \mathbf{a}_2(2)}{\delta_y}\right) < n_2. \tag{8}$$

Here, $\mathrm{round}(a)$ denotes the operation of rounding the element a to the nearest integer. In addition, following we denote some matrix and vector operations which will be used later.

As shown in [14], by applying Yee's finite-difference scheme to the computational cell Ω_c, the discretizations for (1a) can be obtained as the following matrix representations:

$$C\mathbf{e} = \iota\omega\mathbf{b}, \quad C^*\mathbf{h} = -\iota\omega\mathbf{d}, \tag{9}$$

where

$$C = \begin{bmatrix} 0 & -C_3 & C_2 \\ C_3 & 0 & -C_1 \\ -C_2 & C_1 & 0 \end{bmatrix} \tag{10}$$

with

$$C_1 = \frac{1}{\delta_x} I_{n_3} \otimes I_{n_2} \otimes \begin{bmatrix} -I_{n_1-1} & I_{n_1-1} \\ e^{\imath 2\pi \mathbf{k}\cdot\mathbf{a}_1} & -1 \end{bmatrix}, \tag{11a}$$

$$C_2 = \frac{1}{\delta_y} I_{n_3} \otimes \begin{bmatrix} & -I_{n_1(n_2-1)} & \\ & & \\ e^{\imath 2\pi \mathbf{k}\cdot(\rho_1\mathbf{a}_1+\mathbf{a}_2)} & \begin{bmatrix} 0 & e^{-\imath 2\pi \mathbf{k}\cdot\mathbf{a}_1} I_{m_1} \\ I_{n_1-m_1} & 0 \end{bmatrix} & \\ & & \end{bmatrix} \begin{matrix} \\ \\ \end{matrix} \begin{bmatrix} I_{n_1(n_2-1)} \\ \\ -I_{n_1} \end{bmatrix}, \tag{11b}$$

$$C_3 = \frac{1}{\delta_z} \begin{bmatrix} -I_{n_1 n_2(n_3-1)} & I_{n_1 n_2(n_3-1)} \\ e^{\imath 2\pi \mathbf{k}\cdot\mathbf{a}_3} J_3 & -I_{n_1 n_2} \end{bmatrix}. \tag{11c}$$

See [14] or Appendix A for the detailed definitions of the matrix J_3 for all Bravais lattices.

We define

$$\mathbf{f}_{m,\ell} = \begin{bmatrix} 1 & e^{\ell\hat{\theta}_m} & \cdots & e^{(m-1)\ell\hat{\theta}_m} \end{bmatrix}^\top, \quad \hat{\theta}_m = \frac{\imath 2\pi}{m}, \tag{12a}$$

$$\hat{\mathbf{a}}_2 = \mathbf{a}_2 - (\frac{m_1}{n_1} - \rho_1)\mathbf{a}_1, \tag{12b}$$

$$\hat{\mathbf{a}}_3 = \mathbf{a}_3 - \left(\tfrac{m_3}{n_2} - \rho_3\right)\mathbf{a}_2 - \left\{\left(\tfrac{m_3}{n_2} - \rho_3\right)\rho_1 - \rho_2 - \tfrac{m_1}{n_1}\tfrac{m_3}{n_2} + \tfrac{m_2+\rho_3 m_1}{n_1}\right\}\mathbf{a}_1, \tag{12c}$$

$$\theta_{\mathbf{a}_1} = \frac{\imath 2\pi \mathbf{k}\cdot\mathbf{a}_1}{n_1},\ \theta_{\hat{\mathbf{a}}_2,i} = \frac{\imath 2\pi}{n_2}\left(\mathbf{k}\cdot\hat{\mathbf{a}}_2 - \frac{m_1}{n_1} i\right), \tag{12d}$$

$$\theta_{\hat{\mathbf{a}}_3,i,j} = \frac{\imath 2\pi}{n_3}\left\{\mathbf{k}\cdot\hat{\mathbf{a}}_3 - \frac{m_3}{n_2} j + \left(\frac{m_1}{n_1}\frac{m_3}{n_2} - \frac{\rho_3 m_1 + m_2}{n_1}\right) i\right\}, \tag{12e}$$

and

$$D_{\mathbf{a}_1} = \operatorname{diag}\left(1, e^{\theta_{\mathbf{a}_1}}, \cdots, e^{(n_1-1)\theta_{\mathbf{a}_1}}\right), \tag{13a}$$

$$D_{\hat{\mathbf{a}}_2,i} = \operatorname{diag}\left(1, e^{\theta_{\hat{\mathbf{a}}_2,i}}, \cdots, e^{(n_2-1)\theta_{\hat{\mathbf{a}}_2,i}}\right), \tag{13b}$$

$$D_{\hat{\mathbf{a}}_3,i,j} = \operatorname{diag}\left(1, e^{\theta_{\hat{\mathbf{a}}_3,i,j}}, \cdots, e^{(n_3-1)\theta_{\hat{\mathbf{a}}_3,i,j}}\right). \tag{13c}$$

Let

$$\mathbf{x}_i = D_{\mathbf{a}_1}\mathbf{f}_{n_1,i}, \quad \mathbf{y}_{i,j} = D_{\hat{\mathbf{a}}_2,i}\mathbf{f}_{n_2,j}, \quad \mathbf{z}_{i,j,k} = D_{\hat{\mathbf{a}}_3,i,j}\mathbf{f}_{n_3,k},$$

for $i = 0, \ldots, n_1 - 1$, $j = 0, \ldots, n_2 - 1$, and $k = 0, \ldots, n_3 - 1$. Using $\mathbf{x}_i$, $\mathbf{y}_{i,j}$ and $\mathbf{z}_{i,j,k}$, we define an $n \times n$ matrix T as follows:

$$T = \frac{1}{\sqrt{n}}\begin{bmatrix} T_1 & T_2 & \cdots & T_{n_1} \end{bmatrix}, \quad T_i = \begin{bmatrix} T_{i,1} & T_{i,2} & \cdots & T_{i,n_2} \end{bmatrix} \in \mathbb{C}^{n\times(n_2 n_3)}, \tag{14a}$$

$$T_{i,j} = \begin{bmatrix} \mathbf{z}_{i,j,1}\otimes\mathbf{y}_{i,j}\otimes\mathbf{x}_i & \mathbf{z}_{i,j,2}\otimes\mathbf{y}_{i,j}\otimes\mathbf{x}_i & \cdots & \mathbf{z}_{i,j,n_3}\otimes\mathbf{y}_{i,j}\otimes\mathbf{x}_i \end{bmatrix}, \tag{14b}$$

for $i = 1, \cdots, n_1$ and $j = 1, \cdots, n_2$. As shown in [11], T is unitary.

It has been shown in [4, 11, 14] that C_1, C_2 and C_3 are normal and commute with each other. In addition, C_1, C_2 and C_3 can be diagonalized by means of the common unitary matrix T defined in (14).

THEOREM 1 ([4, 11, 14]). *Let C_ℓ ($\ell = 1,2,3$) be defined as shown in (11). Then, C_1, C_2, and C_3 are simultaneously diagonalizable by means of the unitary matrix $T \in \mathbb{C}^{n\times n}$ in the following form:*

$$C_\ell T = T\Lambda_\ell, \quad \ell = 1,2,3, \tag{15}$$

with

$$\Lambda_1 = \Lambda_{n_1} \otimes I_{n_2} \otimes I_{n_3}, \qquad \Lambda_{n_1} = \frac{1}{\delta_x} diag\left(e^{\theta_1} - 1, \cdots, e^{\theta_{n_1}} - 1\right) \in \mathbb{C}^{n_1\times n_1}, \tag{16a}$$

$$\Lambda_2 = \oplus_{i=1}^{n_1}(\Lambda_{i,n_2} \otimes I_{n_3}), \qquad \Lambda_{i,n_2} = \frac{1}{\delta_y} diag\left(e^{\theta_{i,1}} - 1, \cdots, e^{\theta_{i,n_2}} - 1\right) \in \mathbb{C}^{n_2\times n_2}, \tag{16b}$$

$$\Lambda_3 = \oplus_{i=1}^{n_1} \oplus_{j=1}^{n_2} \Lambda_{i,j,n_3}, \quad \Lambda_{i,j,n_3} = \frac{1}{\delta_z} diag\left(e^{\theta_{i,j,1}} - 1, \cdots, e^{\theta_{i,j,n_3}} - 1\right) \in \mathbb{C}^{n_3\times n_3}, \tag{16c}$$

where $\theta_i = i\hat{\theta}_{n_1} + \theta_{\mathbf{a}_1}$, $\theta_{i,j} = j\hat{\theta}_{n_2} + \theta_{\hat{\mathbf{a}}_2,i}$, $\theta_{i,j,k} = k\hat{\theta}_{n_3} + \theta_{\hat{\mathbf{a}}_3,i,j}$.

As shown in [4, 11], we define

$$\begin{bmatrix}\Phi_0 & \Phi_1 & \Phi_2\end{bmatrix} = \begin{bmatrix}\Lambda_1 & \Lambda_q - \Lambda_1\Lambda_s^* & \Lambda_3^* - \Lambda_2^* \\ \Lambda_2 & \Lambda_q - \Lambda_2\Lambda_s^* & \Lambda_1^* - \Lambda_3^* \\ \Lambda_3 & \Lambda_q - \Lambda_3\Lambda_s^* & \Lambda_2^* - \Lambda_1^*\end{bmatrix} \text{diag}\left(\Lambda_q^{-\frac{1}{2}}, \left(3\Lambda_q^2 - \Lambda_q\Lambda_p\right)^{-\frac{1}{2}}, \left(3\Lambda_q - \Lambda_p\right)^{-\frac{1}{2}}\right), \tag{17}$$

where $\Lambda_q = \Lambda_1^*\Lambda_1 + \Lambda_2^*\Lambda_2 + \Lambda_3^*\Lambda_3 \in \mathbb{R}^{n\times n}$ and $\Lambda_p = \Lambda_s\Lambda_s^* \equiv (\Lambda_1 + \Lambda_2 + \Lambda_3)(\Lambda_1 + \Lambda_2 + \Lambda_3)^* \in \mathbb{R}^{n\times n}$. The following important SVD of C is derived in [4], which provides the bases for n-dimensional null space and $2n$-dimensional range space of C and C^*.

THEOREM 2 ([4]). *There exist unitary matrices*

$$V = \begin{bmatrix}V_0 & V_r\end{bmatrix} \equiv (I_3 \otimes T)\left[\begin{array}{c|cc} \Phi_0 & \Phi_1 & \Phi_2 \end{array}\right] \equiv (I_3 \otimes T)\begin{bmatrix}\Phi_0 & \Phi_r\end{bmatrix}, \tag{18a}$$

$$U = \begin{bmatrix}U_0 & U_r\end{bmatrix} \equiv (I_3 \otimes T)\left[\begin{array}{c|cc} \bar{\Phi}_0 & -\bar{\Phi}_2 & \bar{\Phi}_1 \end{array}\right], \tag{18b}$$

where $V_r, U_r \in \mathbb{C}^{3n\times 2n}$, such that C has the following SVD and compact SVD:

$$C = U\, diag(0_{n\times n}, \Lambda_q^{1/2}, \Lambda_q^{1/2})V^* = U_r\Sigma_r V_r^*, \quad \Sigma_r = diag(\Lambda_q^{1/2}, \Lambda_q^{1/2}) \in \mathbb{R}^{2n\times 2n}. \tag{18c}$$

In Algorithm 2, we summarize the processes of constructing Σ_r, Φ_0, Φ_1, and Φ_2 in Theorem 2 and $D_{\mathbf{a}_1}$, $D_{\hat{\mathbf{a}}_2,i}$, and $D_{\hat{\mathbf{a}}_3,i,j}$ ($i = 1,\ldots,n_1$ and $j = 1,\ldots,n_2$) in (13).

4 FFT-BASED MATRIX-VECTOR MULTIPLICATIONS

From the definition of T given in (14), we can see that T is dense and impossible to explicitly store in memory. Hence, at first glance, the SVD of C in (18c) seems only theoretically interesting. Fortunately, novel and efficient FFT-based matrix-vector multiplications have been proposed in [11, 16] to address this problem, meaning that the SVD given in (18c) is now a useful and powerful tool for applications involving 3D electromagnetism.

To introduce these FFT-based multiplications, we define an orthogonal matrix F_m as follows:

$$F_m = \begin{bmatrix}\mathbf{f}_{m,1} & \cdots & \mathbf{f}_{m,m}\end{bmatrix} \in \mathbb{C}^{m\times m},$$

Algorithm 2 [14] Construction of the matrices in the SVD

Input: Computational lattice translation vectors $\mathbf{a}_1$, $\mathbf{a}_2$, $\mathbf{a}_3$, grid numbers n_1, n_2, n_3, wave vector $\mathbf{k}$.
Output: Λ_1, Λ_2, Λ_3, Λ_q, Σ_r, Φ_0, Φ_1, Φ_2, $D_{\mathbf{a}_1}$, $D_{\hat{\mathbf{a}}_2,i}$, and $D_{\hat{\mathbf{a}}_3,i,j}$ for $i = 1,\ldots,n_1$, $j = 1,\ldots,n_2$.

1: Compute ρ_1, ρ_2 and ρ_3 according to (7).
2: Compute $\delta_x = \mathbf{a}_1(1)/n_1$, $\delta_y = \mathbf{a}_2(2)/n_2$ and $\delta_z = \mathbf{a}_3(3)/n_3$.
3: Compute m_1, m_2 and m_3 according to (8).
4: Compute $\hat{\mathbf{a}}_2$, $\hat{\mathbf{a}}_3$, $\hat{\theta}_{n_1}$, $\hat{\theta}_{n_2}$, $\hat{\theta}_{n_3}$, $\theta_{\mathbf{a}_1}$ according to (12) and $\theta_{\hat{\mathbf{a}}_2} = \frac{\imath 2\pi \mathbf{k}\cdot\hat{\mathbf{a}}_2}{n_2}$, $\theta_{\hat{\mathbf{a}}_3} = \frac{\imath 2\pi \mathbf{k}\cdot\hat{\mathbf{a}}_3}{n_3}$.
5: Compute Λ_{n_1}, Λ_{i,n_2} and Λ_{i,j,n_3} according to (16) with $\theta_i = \theta_{\mathbf{a}_1} + i\hat{\theta}_{n_1}$, $\theta_{i,j} = \theta_{\hat{\mathbf{a}}_2} + j\hat{\theta}_{n_2} - \frac{\imath 2\pi}{n_2}\frac{m_1}{n_1} i$, $\theta_{i,j,k} = \theta_{\hat{\mathbf{a}}_3} + k\hat{\theta}_{n_3} + \frac{\imath 2\pi}{n_3}\left\{\left(\frac{m_1}{n_1}\frac{m_3}{n_2} - \frac{\rho_3 m_1 + m_2}{n_1}\right) i - \frac{m_3}{n_2} j\right\}$ for $i = 1,\ldots,n_1$, $j = 1,\ldots,n_2$ and $k = 1,\ldots,n_3$.
6: Compute $\Lambda_1 = \Lambda_{n_1} \otimes I_{n_2} \otimes I_{n_3}$, $\Lambda_2 = \oplus_{i=1}^{n_1}(\Lambda_{i,n_2} \otimes I_{n_3})$, and $\Lambda_3 = \oplus_{i=1}^{n_1}\oplus_{j=1}^{n_2}\Lambda_{i,j,n_3}$.
7: Compute $\Lambda_q = \Lambda_1^*\Lambda_1 + \Lambda_2^*\Lambda_2 + \Lambda_3^*\Lambda_3$, $\Sigma_r = \mathrm{diag}(\Lambda_q^{1/2}, \Lambda_q^{1/2})$, $\Lambda_s = \Lambda_1 + \Lambda_2 + \Lambda_3$ and $\Lambda_p = \Lambda_s\Lambda_s^*$.
8: Compute Φ_0, Φ_1 and Φ_2 according to (17).
9: Compute $D_{\mathbf{a}_1}$, $D_{\hat{\mathbf{a}}_2,i}$ and $D_{\hat{\mathbf{a}}_3,i,j}$ according to (13) with $\theta_{\hat{\mathbf{a}}_2,i} = \theta_{\hat{\mathbf{a}}_2} - \frac{\imath 2\pi}{n_2}\frac{m_1}{n_1} i$, $\theta_{\hat{\mathbf{a}}_3,i,j} = \theta_{\hat{\mathbf{a}}_3} + \frac{\imath 2\pi}{n_3}\left\{\left(\frac{m_1}{n_1}\frac{m_3}{n_2} - \frac{\rho_3 m_1 + m_2}{n_1}\right) i - \frac{m_3}{n_2} j\right\}$ for $i = 1,\ldots,n_1$, $j = 1,\ldots,n_2$.

where $m = n_1$, n_2, or n_3 and $\{f_{m,\ell}\}_{\ell=1}^m$ is given in (12a). In addition, we let

$$G_{\mathbf{y}} \equiv \begin{bmatrix} \mathrm{diag}(D_{\hat{\mathbf{a}}_2,1}) & \cdots & \mathrm{diag}(D_{\hat{\mathbf{a}}_2,n_1}) \end{bmatrix} \in \mathbb{C}^{n_2\times n_1}, \tag{19a}$$

$$G_{\mathbf{z},i} \equiv \begin{bmatrix} \mathrm{diag}(D_{\hat{\mathbf{a}}_3,i,1}) & \cdots & \mathrm{diag}(D_{\hat{\mathbf{a}}_3,i,n_2}) \end{bmatrix} \in \mathbb{C}^{n_3\times n_2}. \tag{19b}$$

Then, based on the structure of T, the matrix-vector multiplications $T\mathbf{q}$ and $T^*\mathbf{p}$ for given vectors $\mathbf{p}$ and $\mathbf{q}$ can be calculated as sequences of entrywise multiplications, diagonal matrix-vector multiplications and one-dimensional FFT operations [11, 16], as shown in Algorithms 3 and 4.

Algorithm 3 [16] Parallel FFT-based matrix-vector multiplication for $T\mathbf{q}$

Input: $D_{\mathbf{a}_1}$, $G_{\mathbf{y}}$, $G_{\mathbf{z},i}$ for $i = 1,\ldots,n_1$, and any vector $\mathbf{q} = \begin{bmatrix}\mathbf{q}_1^\top & \cdots & \mathbf{q}_{n_1}^\top\end{bmatrix}^\top \in \mathbb{C}^n$ with $\mathbf{q}_i = \begin{bmatrix}\mathbf{q}_{i,1}^\top & \cdots & \mathbf{q}_{i,n_2}^\top\end{bmatrix}^\top$ and $\mathbf{q}_{i,j} \in \mathbb{C}^{n_3}$ for $i = 1,\ldots,n_1, j = 1,\ldots,n_2$.
Output: The vector $\tilde{\mathbf{q}} \equiv T\mathbf{q}$.

1: Set $\widetilde{Q}_{\mathbf{z},i} = \begin{bmatrix}\mathbf{q}_{i,1} & \cdots & \mathbf{q}_{i,n_2}\end{bmatrix} \in \mathbb{C}^{n_3\times n_2}$, for $i = 1,\ldots,n_1$.
2: Compute $\widetilde{Q}_{u\mathbf{z}} = F_{n_3}\begin{bmatrix}\widetilde{Q}_{\mathbf{z},1} & \cdots & \widetilde{Q}_{\mathbf{z},n_1}\end{bmatrix} \in \mathbb{C}^{n_3\times n_1 n_2}$ by 1D backward FFT .
3: Compute $\widetilde{Q}_{\mathbf{z},i} = G_{\mathbf{z},i} \odot \widetilde{Q}_{u\mathbf{z}}(:, (i-1)n_2 + 1 : in_2)$ for $i = 1,\ldots,n_1$.
4: Set $\widetilde{Q}_{\mathbf{y}} = \begin{bmatrix}\widetilde{Q}_{\mathbf{z},1}^\top & \cdots & \widetilde{Q}_{\mathbf{z},n_1}^\top\end{bmatrix} \in \mathbb{C}^{n_2\times n_1 n_3}$.
5: Compute $\widetilde{Q}_{u\mathbf{y}} \equiv \begin{bmatrix}\widetilde{Q}_{u\mathbf{y}}^{(1)} & \cdots & \widetilde{Q}_{u\mathbf{y}}^{(n_1)}\end{bmatrix} = F_{n_2}\widetilde{Q}_{\mathbf{y}}$ by 1D backward FFT .
6: Compute $\widetilde{Q}_{\mathbf{y},k} = G_{\mathbf{y}} \odot \begin{bmatrix}\widetilde{Q}_{u\mathbf{y}}^{(1)}(:,k) & \cdots & \widetilde{Q}_{u\mathbf{y}}^{(n_1)}(:,k)\end{bmatrix}$ for $k = 1,\ldots,n_3$.
7: Set $\widetilde{Q}_{\mathbf{x}} = \begin{bmatrix}\widetilde{Q}_{\mathbf{y},1}^\top & \cdots & \widetilde{Q}_{\mathbf{y},n_3}^\top\end{bmatrix}$.
8: Compute $\widetilde{Q}_{u\mathbf{x}} = F_{n_1}\widetilde{Q}_{\mathbf{x}}$ by 1D backward FFT .
9: Compute $\tilde{\mathbf{q}} = \frac{1}{\sqrt{n_1 n_2 n_3}}\mathrm{vec}(D_{\mathbf{a}_1}\widetilde{Q}_{u\mathbf{x}})$.

Algorithm 4 [16] Parallel FFT-based matrix-vector multiplication for $T^*\mathbf{p}$

Input: $D_{\mathbf{a}_1}$, $D_{\hat{\mathbf{a}}_2,i}$, $G_{\mathbf{z},i}$ for $i = 1,\ldots,n_1$, and any vector $\mathbf{p} = \begin{bmatrix}\mathbf{p}_1^\top & \cdots & \mathbf{p}_{n_3}^\top\end{bmatrix}^\top \in \mathbb{C}^n$ with $\mathbf{p}_k = \begin{bmatrix}\mathbf{p}_{1,k}^\top & \cdots & \mathbf{p}_{n_2,k}^\top\end{bmatrix}^\top$ and $\mathbf{p}_{j,k} \in \mathbb{C}^{n_1}$ for $j = 1,\ldots,n_2$, $k = 1,\ldots,n_3$.
Output: The vector $\tilde{\mathbf{p}} \equiv T^*\mathbf{p}$.
1: Set $\widetilde{P}_{\mathbf{x},k} = \begin{bmatrix}\mathbf{p}_{1,k} & \cdots & \mathbf{p}_{n_2,k}\end{bmatrix} \in \mathbb{C}^{n_1\times n_2}$, for $k = 1,\ldots,n_3$.
2: Compute $\widetilde{P}_{e\mathbf{x}} = D_{\mathbf{a}_1}^* \begin{bmatrix}\widetilde{P}_{\mathbf{x},1} & \cdots & \widetilde{P}_{\mathbf{x},n_3}\end{bmatrix}$.
3: Compute $\widetilde{P}_{u\mathbf{x}} = F_{n_1}^* \widetilde{P}_{e\mathbf{x}} \in \mathbb{C}^{n_1\times n_2 n_3}$ by 1D forward FFT .
4: Set $\widetilde{P}_{u\mathbf{x}}^{(i)} = \begin{bmatrix}\widetilde{P}_{u\mathbf{x}}(i,1:n_2)^\top & \cdots & \widetilde{P}_{u\mathbf{x}}(i,(n_3-1)n_2+1:n_3n_2)^\top\end{bmatrix}$, for $i = 1,\ldots,n_1$.
5: Compute $\widetilde{P}_{e\mathbf{y}} = \begin{bmatrix}D_{\hat{\mathbf{a}}_2,1}^* \widetilde{P}_{u\mathbf{x}}^{(1)} & \cdots & D_{\hat{\mathbf{a}}_2,n_1}^* \widetilde{P}_{u\mathbf{x}}^{(n_1)}\end{bmatrix}$.
6: Compute $\widetilde{P}_{u\mathbf{y}} \equiv \begin{bmatrix}\widetilde{P}_{u\mathbf{y}}^{(1)} & \cdots & \widetilde{P}_{u\mathbf{y}}^{(n_1)}\end{bmatrix} = F_{n_2}^* \widetilde{P}_{e\mathbf{y}} \in \mathbb{C}^{n_2\times n_1 n_3}$ by 1D forward FFT .
7: Compute $\widetilde{P}_{e\mathbf{z}} = \begin{bmatrix}\bar{G}_{\mathbf{z},1} \odot (\widetilde{P}_{u\mathbf{y}}^{(1)})^\top & \cdots & \bar{G}_{\mathbf{z},n_1} \odot (\widetilde{P}_{u\mathbf{y}}^{(n_1)})^\top\end{bmatrix}$.
8: Compute $\tilde{\mathbf{p}} = \frac{1}{\sqrt{n_1 n_2 n_3}} \mathrm{vec}(F_{n_3}^* \widetilde{P}_{e\mathbf{z}})$ by 1D forward FFT.

4.1 Comparison with 3D FFTs

In this subsection, we show that Algorithms 3 and 4 can be related to the standard 3D FFTs $\frac{1}{\sqrt{n_1 n_2 n_3}} (F_{n_3} \otimes F_{n_2} \otimes F_{n_1})\,\mathbf{q}$ and $\frac{1}{\sqrt{n_1 n_2 n_3}} (F_{n_3} \otimes F_{n_2} \otimes F_{n_1})^*\,\mathbf{p}$, respectively.

The details can be derived as follows. If we reshape $\mathbf{p}$ as a matrix $P \in \mathbb{C}^{n_1\times n_2 n_3}$ with $\mathrm{vec}(P) = \mathbf{p}$ and let

$$P_{\mathbf{x}} \equiv \begin{bmatrix}p_{\mathbf{x},1}^\top & \cdots & p_{\mathbf{x},n_1}^\top\end{bmatrix}^\top = F_{n_1}^* P \in \mathbb{C}^{n_1\times n_2 n_3}, \tag{20a}$$

$$P_{\mathbf{x},i} \in \mathbb{C}^{n_2\times n_3} \text{ with } \mathrm{vec}(P_{\mathbf{x},i}) = p_{\mathbf{x},i}^\top,\ i = 1,\ldots,n_1, \tag{20b}$$

$$P_{\mathbf{y}} \equiv \begin{bmatrix}P_{\mathbf{y},1} & \cdots & P_{\mathbf{y},n_1}\end{bmatrix} = F_{n_2}^* \begin{bmatrix}P_{\mathbf{x},1} & \cdots & P_{\mathbf{x},n_1}\end{bmatrix} \in \mathbb{C}^{n_2\times n_1 n_3}, \tag{20c}$$

$$P_{\mathbf{z}} \equiv \begin{bmatrix}P_{\mathbf{z},1} & \cdots & P_{\mathbf{z},n_1}\end{bmatrix} = F_{n_3}^* \begin{bmatrix}P_{\mathbf{y},1}^\top & \cdots & P_{\mathbf{y},n_1}^\top\end{bmatrix} \in \mathbb{C}^{n_3\times n_1 n_2}, \tag{20d}$$

then, from the properties of the Kronecker product, namely,

$$(B^\top \otimes A)\mathrm{vec}(X) = \mathrm{vec}(AXB), \quad (A\otimes B)^* = A^* \otimes B^*, \tag{21}$$

it follows that

$$\begin{aligned}(F_{n_3}\otimes F_{n_2}\otimes F_{n_1})^*\,\mathbf{p} &= \mathrm{vec}\left(F_{n_1}^* P \left(\bar{F}_{n_3}\otimes \bar{F}_{n_2}\right)\right) = \mathrm{vec}\left(\begin{bmatrix}(F_{n_3}^*\otimes F_{n_2}^*)p_{\mathbf{x},1}^\top & \cdots & (F_{n_3}^*\otimes F_{n_2}^*)p_{\mathbf{x},n_1}^\top\end{bmatrix}^\top\right)\\ &= \mathrm{vec}\left(\begin{bmatrix}\mathrm{vec}(F_{n_2}^* P_{\mathbf{x},1}\bar{F}_{n_3}) & \cdots & \mathrm{vec}(F_{n_2}^* P_{\mathbf{x},n_1}\bar{F}_{n_3})\end{bmatrix}^\top\right) = \mathrm{vec}\left(\begin{bmatrix}\mathrm{vec}(P_{\mathbf{y},1}\bar{F}_{n_3}) & \cdots & \mathrm{vec}(P_{\mathbf{y},n_1}\bar{F}_{n_3})\end{bmatrix}^\top\right)\\ &= \mathrm{vec}\left(\begin{bmatrix}\mathrm{vec}(P_{\mathbf{z},1}^\top) & \cdots & \mathrm{vec}(P_{\mathbf{z},n_1}^\top)\end{bmatrix}^\top\right).\end{aligned} \tag{22}$$

Let $D_{\mathbf{a}_1}$ and $D_{\hat{\mathbf{a}}_2,i}$, $i = 1,\ldots,n_1$, in Lines 2 and 5 of Algorithm 4 be identity matrices of appropriate dimensions. Then, $\widetilde{P}_{e\mathbf{x}}$ and $\widetilde{P}_{e\mathbf{y}}$ in Lines 2 and 5 are equal to $P_{\mathbf{x}}$ in (20a) and $P_{\mathbf{y}}$ in (20c), respectively. Moreover, if $D_{\hat{\mathbf{a}}_3,i,j}$, $i = 1,\ldots,n_1, j = 1,\ldots,n_2$, are $n_3\times n_3$ identity matrices, i.e., $G_{\mathbf{z},i} = \mathbf{1}_{n_3}\mathbf{1}_{n_2}^\top$, $i = 1,\ldots,n_1$, in Line 7 of Algorithm 4, then $F_{n_3}^*\widetilde{P}_{e\mathbf{z}}$ in Line 8 of Algorithm 4 is equal to $P_{\mathbf{z}}$ in (20d). Here, $\mathbf{1}_m$ is an $m\times 1$ vector with each component being 1. This means that Algorithm 4 is an 3D forward FFT in (22) if $D_{\mathbf{a}_1}$, $D_{\hat{\mathbf{a}}_2,i}$ and $D_{\hat{\mathbf{a}}_3,i,j}$ are identity matrices of appropriate dimensions.

Meanwhile, we can reshape the vector $\mathbf{q}$ as a matrix $Q \in \mathbb{C}^{n_1 n_2 \times n_3}$ with $\mathrm{vec}(Q) = \mathbf{q}$ and let

$$Q_{\mathbf{z}} \equiv [\tilde{q}_1, \cdots, \tilde{q}_{n_3}] = Q F_{n_3}^{\top} = (F_{n_3} Q^{\top})^{\top} \in \mathbb{C}^{n_1 n_2 \times n_3}, \tag{23a}$$

$$Q_{\mathbf{z},k} \in \mathbb{C}^{n_1 \times n_2} \text{ with } \mathrm{vec}(Q_{\mathbf{z},k}) = \tilde{q}_k, \ k = 1, \ldots, n_3, \tag{23b}$$

$$Q_{\mathbf{y}} \equiv \begin{bmatrix} Q_{\mathbf{y},1} & \cdots & Q_{\mathbf{y},n_3} \end{bmatrix} = F_{n_2} \begin{bmatrix} Q_{\mathbf{z},1}^{\top} & \cdots & Q_{\mathbf{z},n_3}^{\top} \end{bmatrix} \in \mathbb{C}^{n_2 \times n_1 n_3}, \tag{23c}$$

$$Q_{\mathbf{x}} \equiv \begin{bmatrix} Q_{\mathbf{x},1} & \cdots & Q_{\mathbf{x},n_3} \end{bmatrix} = F_{n_1} \begin{bmatrix} Q_{\mathbf{y},1}^{\top} & \cdots & Q_{\mathbf{y},n_3}^{\top} \end{bmatrix} \in \mathbb{C}^{n_1 \times n_2 n_3}. \tag{23d}$$

Then, using the properties of the Kronecker product given in (21), we obtain

$$(F_{n_3} \otimes F_{n_2} \otimes F_{n_1})\, \mathbf{q} = \mathrm{vec}(Q_{\mathbf{x}}). \tag{24}$$

Using a similar discussion to the above paragraph, Algorithm 3 can be simplified to the 3D backward FFT in (23) if $D_{\hat{\mathbf{a}}_1}$, $D_{\hat{\mathbf{a}}_2,i}$ and $D_{\hat{\mathbf{a}}_3,i,j}$ are identity matrices of appropriate dimensions.

4.2 Implementation

In Algorithms 3 and 4, the complexities without parallelization for computing $T\mathbf{q}$ and $T^*\mathbf{p}$ are both equal to

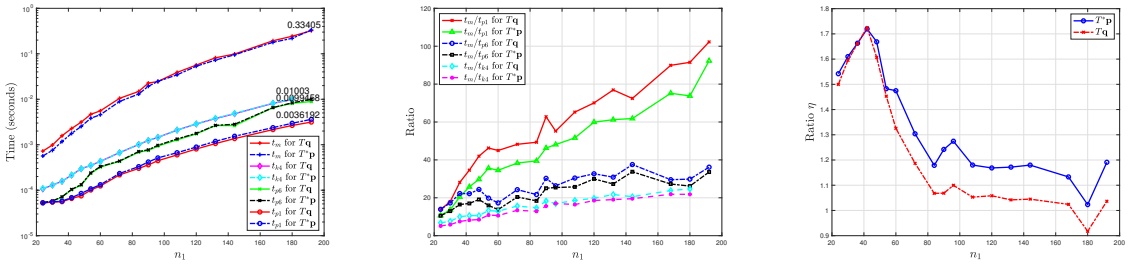
$$c = 4n + (n_1 n_2 \mathcal{O}(n_3 \log n_3) + n_1 n_3 \mathcal{O}(n_2 \log n_2) + n_2 n_3 \mathcal{O}(n_1 \log n_1)),$$

and more details can be found in Appendix B. The highest computational costs for $T\mathbf{q}$ and $T^*\mathbf{p}$ are those of the matrix computations in Steps 2, 5 and 8 of Algorithm 3 and in Steps 3, 6 and 8 of Algorithm 4, respectively. However, the corresponding matrix in each of these steps can be computed by means of m 1D FFTs, where $m = n_1 n_2$, $n_1 n_3$, or $n_2 n_3$. For example, in Step 2 of Algorithm 3, $\widetilde{Q}_{u\mathbf{z}}$ is computed by means of $n_1 n_2$ 1D backward FFTs involving F_{n_3}. These $n_1 n_2$ 1D backward FFTs can be performed in parallel using multicore/many-core CPU and GPU computations. In this case, there is no need to consider any parallelization for the 1D FFTs. Therefore, highly efficient performance can be achieved for Algorithms 3 and 4 [16].

By contrast, in the MATLAB environment, the 1D backward FFTs in Steps 2, 5 and 8 of Algorithm 3 are computed by means of `ifft`(Q), where the matrices Q in these steps are $n_3 \times n_1 n_2$, $n_2 \times n_1 n_3$ and $n_1 \times n_2 n_3$ matrices, respectively. Similarly, the 1D forward FFTs in Steps 3, 6 and 8 of Algorithm 4 are computed by means of `fft`(P), where the matrices P are $n_1 \times n_2 n_3$, $n_2 \times n_1 n_3$ and $n_3 \times n_1 n_2$ matrices, respectively. On an NVIDIA GPU architecture, `cufftExecZ2Z(*,*,*,CUFFT_INVERSE)` and `cufftExecZ2Z(*,*,*,CUFFT_FORWARD)` in the CUDA Toolkit, corresponding to `ifft`(Q) and `fft`(P), respectively, are used to compute the 1D backward and forward FFTs in Algorithms 3 and 4. Such 1D backward and forward FFTs can be performed in parallel using multiple GPU threads. More details of the possible implementations of Algorithms 3 and 4 for improved performance can be found in [16].

4.3 Performance of Algorithms 3 and 4

In this section, we illustrate the efficiency of Algorithms 3 and 4 in the MATLAB environment and on a single NVIDIA GPU architecture. We performed numerical experiments on a workstation with two Intel Xeon Gold 6132 (3.2 GHz, 8-core) CPUs and 128 GB of main memory running the Red Hat Enterprise Linux operating system. For the GPU architecture, we used the NVIDIA Quadro P6000 GPU with 24 GB of GDDR5X memory, the Tesla K40 GPU with 12 GB of GDDR5 memory, and the Tesla P100 GPU



(a) Timing performance (b) Timing ratios (c) Timing ratios η

Fig. 2. Timing performance for computing $T\mathbf{q}$ and $T^*\mathbf{p}$ for various $n = n_1^3$. Here, $n_1^3 = 192^3 = 7,077,888$.

with 16 GB of GDDR5 memory, each on the Linux operating system. Let t_m, t_{p6}, t_{k8} and t_{p1} denote the computation times when using MATLAB and the P6000, K40 and P100 GPUs, respectively. Figure 2(a) shows the corresponding timing results for Algorithms 3 and 4 with $n_1 = n_2 = n_3$. These results show that Algorithms 3 and 4 are highly efficient on the different devices. It takes less than $t_m = 3.3 \times 10^{-1}$, $t_{p6} = 9.9 \times 10^{-3}$, $t_{k8} = 1.0 \times 10^{-2}$, and $t_{p1} = 3.6 \times 10^{-3}$ seconds, respectively, to compute a matrix-vector multiplication $T^*\mathbf{p}$ or $T\mathbf{q}$ even for the matrix T of dimension $192^3 = 7,077,888$. In Figure 2(b), we present the ratios t_m/t_{p1}, t_m/t_{p6} and t_m/t_{k8} for computing $T\mathbf{q}$ and $T^*\mathbf{p}$. These ratios gradually increase as the dimension $n = n_1^3$ increases, and $t_m/t_{p1} \approx 100$ for $n_1 = 192$, which shows that the many-core architecture in a GPU can take advantage of the parallel computation of many 1D backward and 1D forward FFTs.

On an NVIDIA GPU architecture, we can apply `cufftExecZ2Z` and `cufftPlan3d` in the CUDA Toolkit, namely, 3D FFTs, to efficiently compute $\mathbf{p}_{3d} \equiv (F_{n_3} \otimes F_{n_2} \otimes F_{n_1})^* \mathbf{p}$ and $\mathbf{q}_{3d} \equiv (F_{n_3} \otimes F_{n_2} \otimes F_{n_1})\, \mathbf{q}$. Let t_{3d} denote the computation time required to compute $\mathbf{p}_{3d}$ and $\mathbf{q}_{3d}$ using 3D FFTs with the Tesla P100 GPU. Figure 2(c) shows the timing ratios $\eta = t_{p1}/t_{3d}$ with various $n_1 = n_2 = n_3$ for computing $(T^*\mathbf{p}, (F_{n_3} \otimes F_{n_2} \otimes F_{n_1})^* \mathbf{p})$ and $(T\mathbf{q}, (F_{n_3} \otimes F_{n_2} \otimes F_{n_1})\, \mathbf{q})$. These ratios are less than or equal to 1.3 and 1.1 for $T^*\mathbf{p}$ and $T\mathbf{q}$, respectively, when $n_1 > 80$. These findings show that the efficiency of Algorithms 3 and 4 is close to that of the 3D backward and forward FFTs in the CUDA Toolkit.

5 SIMULATIONS OF THE BAND STRUCTURES OF 3D NONDISPERSIVE PCS

For isotropic media, the matrix representations of (9) for the 3D Maxwell's equations in (1a) can be rewritten as a GEP:

$$\begin{bmatrix} 0 & -C^* \\ C & 0 \end{bmatrix} \begin{bmatrix} \mathbf{e} \\ \mathbf{h} \end{bmatrix} = \iota\omega \begin{bmatrix} B_\varepsilon & B_\xi \\ B_\zeta & B_\mu \end{bmatrix} \begin{bmatrix} \mathbf{e} \\ \mathbf{h} \end{bmatrix}, \tag{25}$$

where B_ε, B_μ, B_ξ and B_ζ are diagonal matrices that are the discrete counterparts of ε, μ, ξ and ζ, respectively. Based on Theorems 1 and 2 along with Algorithms 3 and 4, an efficient preconditioning scheme [13, 15] is proposed to solve such linear systems for 3D nondispersive/dispersive PCs (i.e., with $\zeta = \xi = 0$ and $B_\xi = B_\zeta = 0$), and null-space free techniques are developed for simulating the band structures of 3D nondispersive PCs [11] and complex media [4]. In this section, we focus on nondispersive PCs with $\mu = 1$ (i.e., $B_\mu = I$) and a constant isotropic permittivity ε (i.e., B_ε is diagonal and positive definite). The GEP in

(25) can be simplified to $C\mathbf{e} = \iota\omega\mathbf{h}$ and $C^*\mathbf{h} = -\iota\omega B_\varepsilon \mathbf{e}$, which implies that

$$C^*C\mathbf{e} = \lambda B_\varepsilon \mathbf{e} \tag{26a}$$

and

$$CB_\varepsilon^{-1}C^*\mathbf{h} = \lambda\mathbf{h}, \tag{26b}$$

where $\lambda = \omega^2$.

5.1 Generation of the matrices B_ε for all 14 Bravais lattices

In this subsection, we will introduce the composition of the diagonal matrix B_ε. Discretizing $\nabla \times H = -\iota\omega\varepsilon E$ in (1a) at the point $\mathbf{r}_c \in \Omega_c$ by means of Yee's finite-difference scheme, we take the corresponding diagonal element $b_{\ell\ell}$ of B_ε to be

$$b_{\ell\ell} = \begin{cases} \varepsilon_{in}, & \text{if } \mathbf{r}_c \in \text{ material structure,} \\ \varepsilon_o, & \text{otherwise,} \end{cases} \tag{27}$$

where ε_{in} and ε_o are the electric and vacuum permittivities, respectively. Therefore, the composition of B_ε depends on the geometric structure of the dielectric material. As shown in [26] or Figure 1(a), all geometric structures are defined in terms of a primitive cell $\Omega_p = \{\sum_{\ell=1}^3 \alpha_\ell \tilde{\mathbf{a}}_\ell; \alpha_1, \alpha_2, \alpha_3 \in [0,1]\}$. We need to transform the discrete point $\mathbf{r}_c \in \Omega_c$ into a point $\mathbf{r}_p \in \Omega_p$ and then determine the associated value of $b_{\ell\ell}$ in (27) depending on whether $\mathbf{r}_p$ lies in the material structure. For any $\mathbf{r}_c \in \Omega_c$, from (5), we have

$$\mathbf{r}_c = \begin{bmatrix} \mathbf{a}_1 & \mathbf{a}_2 & \mathbf{a}_3 \end{bmatrix} \mathbf{c} = Q^\top \begin{bmatrix} \tilde{\mathbf{a}}_1 & \tilde{\mathbf{a}}_2 & \tilde{\mathbf{a}}_3 \end{bmatrix} \Pi\mathbf{c} \equiv Q^\top \begin{bmatrix} \tilde{\mathbf{a}}_1 & \tilde{\mathbf{a}}_2 & \tilde{\mathbf{a}}_3 \end{bmatrix} \tilde{\mathbf{c}},$$

where $\tilde{\mathbf{c}} \equiv \Pi\mathbf{c} = \Pi \begin{bmatrix} \mathbf{a}_1 & \mathbf{a}_2 & \mathbf{a}_3 \end{bmatrix}^{-1} \mathbf{r}_c$. Based on the periodicity of the geometric structure, we can move the vector $\begin{bmatrix} \tilde{\mathbf{a}}_1 & \tilde{\mathbf{a}}_2 & \tilde{\mathbf{a}}_3 \end{bmatrix} \tilde{\mathbf{c}}$ along the directions $\tilde{\mathbf{a}}_1$, $\tilde{\mathbf{a}}_2$, and $\tilde{\mathbf{a}}_3$ into Ω_p as follows:

$$\mathbf{r}_p = \begin{bmatrix} \tilde{\mathbf{a}}_1 & \tilde{\mathbf{a}}_2 & \tilde{\mathbf{a}}_3 \end{bmatrix} (\tilde{\mathbf{c}} - \lfloor \tilde{\mathbf{c}} \rfloor) \in \Omega_p,$$

where $\lfloor \tilde{\mathbf{c}} \rfloor$ denotes the application of the floor operator to $\tilde{\mathbf{c}}$. Let $\Omega_{p,m}$ denote the material structure in Ω_p. Then, $b_{\ell\ell}$ in (27) can be written as

$$b_{\ell\ell} = \begin{cases} \varepsilon_{in} & \text{if } \mathbf{r}_p \in \Omega_{p,m}, \\ \varepsilon_o & \text{if } \mathbf{r}_p \in \Omega_{p,o} \equiv \Omega_p \backslash \Omega_{p,m}. \end{cases} \tag{28}$$

Therefore, we can apply Algorithms 5 and 6 to construct the diagonal matrix B_ε when the material domain $\Omega_{p,m}$ is given.

Regarding the construction of the material domain $\Omega_{p,m}$, we can systematically generate some such domains. The crystal structures in [26] are constructed as combinations of spheres $\mathcal{S}_1, \ldots, \mathcal{S}_s$ and cylinders $\mathcal{C}_1, \ldots, \mathcal{C}_t$ (see, e.g., Figure 3). The center of $\mathcal{S}_j$ is given by $\begin{bmatrix} \tilde{\mathbf{a}}_1 & \tilde{\mathbf{a}}_2 & \tilde{\mathbf{a}}_3 \end{bmatrix} \tilde{\mathbf{s}}_j$, and the cylinder $\mathcal{C}_k$ is defined by the centers of its top and bottom circles, denoted by $\tilde{\mathbf{o}}_{t,k} \equiv \begin{bmatrix} \tilde{\mathbf{a}}_1 & \tilde{\mathbf{a}}_2 & \tilde{\mathbf{a}}_3 \end{bmatrix} \tilde{\mathbf{c}}_{t,k}$ and $\tilde{\mathbf{o}}_{b,k} \equiv \begin{bmatrix} \tilde{\mathbf{a}}_1 & \tilde{\mathbf{a}}_2 & \tilde{\mathbf{a}}_3 \end{bmatrix} \tilde{\mathbf{c}}_{b,k}$, respectively. Thus, the geometric structure of a dielectric material will be determined when the radii

Algorithm 5 Construction of the ℓth diagonal element of B_ε

Input: Lattice translation vectors $\tilde{\mathbf{a}}_1$, $\tilde{\mathbf{a}}_2$, $\tilde{\mathbf{a}}_3$, computational lattice translation vectors $\mathbf{a}_1$, $\mathbf{a}_2$, $\mathbf{a}_3$, electric and vacuum permittivities ε_{in} and ε_o, grid point $\mathbf{r}_c$, index ℓ, material domain $\Omega_{p,m}$, and permutation matrix Π.

Output: Element $B_\varepsilon(\ell,\ell)$

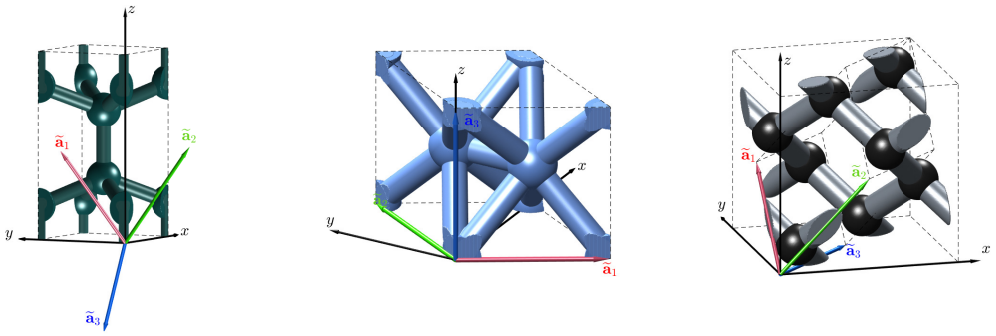
Compute $\tilde{\mathbf{c}} = \Pi\,[\mathbf{a}_1 \;\; \mathbf{a}_2 \;\; \mathbf{a}_3]^{-1}\,\mathbf{r}_c$ and $\mathbf{r}_p = \left[\tilde{\mathbf{a}}_1 \;\; \tilde{\mathbf{a}}_2 \;\; \tilde{\mathbf{a}}_3\right](\tilde{\mathbf{c}} - \lfloor\tilde{\mathbf{c}}\rfloor)$.
if $\mathbf{r}_p \in \Omega_{p,m}$ **then**
  Set $B_\varepsilon(\ell,\ell) = \varepsilon_{in}$.
else
  Set $B_\varepsilon(\ell,\ell) = \varepsilon_o$.
end if

Algorithm 6 Construction of B_ε

Input: Lattice translation vectors $\tilde{\mathbf{a}}_1$, $\tilde{\mathbf{a}}_2$, $\tilde{\mathbf{a}}_3$, computational lattice translation vectors $\mathbf{a}_1$, $\mathbf{a}_2$, $\mathbf{a}_3$, electric and vacuum permittivities ε_{in} and ε_o, material domain $\Omega_{p,m}$, and permutation matrix Π, grid numbers n_1, n_2, n_3.

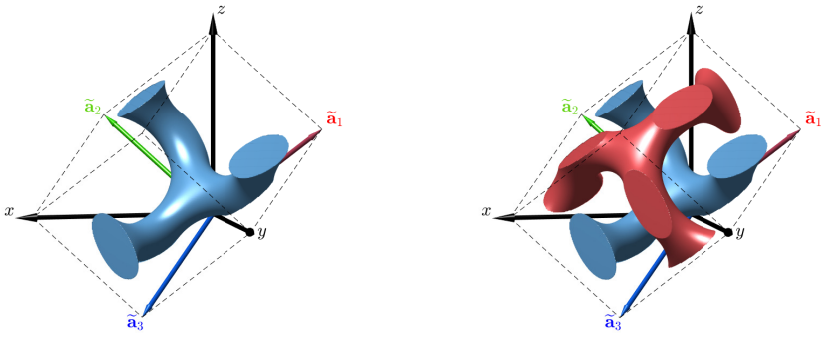
Output: Diagonal matrix B_ε.

Compute $\delta_x = \mathbf{a}_1(1)/n_1$, $\delta_y = \mathbf{a}_2(2)/n_2$ and $\delta_z = \mathbf{a}_3(3)/n_3$.
for $k = 0, \ldots, n_3 - 1$ **do**
  for $j = 0, \ldots, n_2 - 1$ **do**
    for $i = 0, \ldots, n_1 - 1$ **do**
      Set $\tilde{\ell} = (i+1) + j \times n_1 + k \times n_1 \times n_2$.
      Call Algorithm 5 with $\mathbf{r}_c = [(i - \frac{1}{2})\delta_x, j\delta_y, k\delta_z]^\top$ and $\ell = \tilde{\ell}$ to determine the value of $B_\varepsilon(\ell,\ell)$.
      Call Algorithm 5 with $\mathbf{r}_c = [i\delta_x, (j - \frac{1}{2})\delta_y, k\delta_z]^\top$ and $\ell = n + \tilde{\ell}$ to determine the value of $B_\varepsilon(\ell,\ell)$.
      Call Algorithm 5 with $\mathbf{r}_c = [i\delta_x, j\delta_y, (k - \frac{1}{2})\delta_z]^\top$ and $\ell = 2n + \tilde{\ell}$ to determine the value of $B_\varepsilon(\ell,\ell)$.
    end for
  end for
end for



(a) Body-centered tetragonal A_tI4_139_4 (b) Hexagonal AB2_hP3_191_a_d (c) Face-centered cubic A_cF8_227_a

Fig. 3. Three crystal structures in [26].



(a) Single gyroid (b) Double gyroid

Fig. 4. Primitive cells for (a) a single-gyroid PC and (b) a double-gyroid PC.

$\{\rho_1, \ldots, \rho_s\}$ and $\{\gamma_1, \ldots, \gamma_t\}$ of the spheres and cylinders, respectively, are given, i.e.,

$$\mathcal{S}_j = \left\{\tilde{\mathbf{r}} \mid \| \begin{bmatrix} \tilde{\mathbf{a}}_1 & \tilde{\mathbf{a}}_2 & \tilde{\mathbf{a}}_3 \end{bmatrix} \tilde{\mathbf{s}}_j - \tilde{\mathbf{r}} \|_2 \leq \rho_j \right\}, \ j = 1, \ldots, s, \tag{29a}$$

$$\mathcal{C}_k = \{\tilde{\mathbf{r}} \mid \mathbf{l}_k^*(\tilde{\mathbf{r}} - \tilde{\mathbf{o}}_{b,k}) \geq 0, \mathbf{l}_k^*(\tilde{\mathbf{r}} - \tilde{\mathbf{o}}_{t,k}) \leq 0, \|(\tilde{\mathbf{r}} - \tilde{\mathbf{o}}_{b,k}) \times \mathbf{l}_k\|_2 \leq \gamma_k, \mathbf{l}_k = \tilde{\mathbf{o}}_{b,k} - \tilde{\mathbf{o}}_{t,k}\}, \ k = 1, \ldots, t. \tag{29b}$$

This means that $\Omega_{p,m}$ can be defined as

$$\Omega_{p,m} = \left(\cup_{j=1}^{s} \mathcal{S}_j\right) \cup \left(\cup_{k=1}^{t} \mathcal{C}_k\right) \cap \Omega_p.$$

The construction of $\Omega_{p,m}$ as defined in [26] is described in Algorithm 7.

Algorithm 7 Construction of $\Omega_{p,m}$ as defined in [26]

Input: Lattice translation vectors $\tilde{\mathbf{a}}_1$, $\tilde{\mathbf{a}}_2$, $\tilde{\mathbf{a}}_3$, center parameter vector $\tilde{\mathbf{s}}_j$ of the sphere $\mathcal{S}_j$, $j = 1, \ldots, s$, center parameter vectors $\tilde{\mathbf{c}}_{t,k}$ and $\tilde{\mathbf{c}}_{b,k}$ of the cylinder $\mathcal{C}_k$, $k = 1, \ldots, t$.
Output: Crystal domain $\Omega_{p,m}$.
1: Set the values of the radius ρ_j of the sphere $\mathcal{S}_j$, $j = 1, \ldots, s$, and the radius γ_k of the cylinder $\mathcal{C}_k$, $k = 1, \ldots, t$.
2: Construct $\mathcal{S}_j$ for $j = 1, \ldots, s$ and $\mathcal{C}_k$ for $k = 1, \ldots, t$ defined in (29).
3: Construct $\Omega_{p,m}$ as $\Omega_{p,m} = \left(\cup_{j=1}^{s} \mathcal{S}_j\right) \cup \left(\cup_{k=1}^{t} \mathcal{C}_k\right)$.

In addition to the crystal structures in [26], there are many other crystal structures that are determined by different isosurfaces [24, 33, 35–37]. For example, the structure of a gyroid PC [24, 33] is a body-centered cubic lattice in a cubic system. The single- and double-gyroid surfaces shown in Figure 4 can be approximated by the sets $\Omega_{p,m} := \{\mathbf{r} \in \mathbb{R}^3; g(\mathbf{r}) > 1.1\}$ and $\Omega_{p,m} := \{\mathbf{r} \in \mathbb{R}^3; g(\mathbf{r}) > 1.1\} \cup \{\mathbf{r} \in \mathbb{R}^3; g(-\mathbf{r}) > 1.1\}$, respectively, where $\mathbf{r} = (x, y, z)$ and

$$g(\mathbf{r}) = \sin(2\pi x/a)\cos(2\pi y/a) + \sin(2\pi y/a)\cos(2\pi z/a) + \sin(2\pi z/a)\cos(2\pi x/a).$$

Based on a given $\Omega_{p,m}$, we can easily use Algorithm 6 to generate the associated diagonal matrix B_ε.

5.2 Efficient preconditioner for solving linear systems

The GEP expressed in (26) can be solved using the Krylov method, the Jacobi-Davidson method, or the inexact shift-invert residual Arnoldi method [13]. In each iteration of these methods, it is necessary to solve a linear system of the form

$$(C^*C - \sigma B_\varepsilon)\,\mathbf{x} = \mathbf{b} \tag{30a}$$

or

$$\left(CB_\varepsilon^{-1}C^* - \sigma I\right)\mathbf{y} = \mathbf{b} \tag{30b}$$

for a given shift σ and vector $\mathbf{b}$. A major computational issue is how to choose efficient preconditioners for solving (30), which is not a trivial task. None of the traditional preconditioners, such as symmetric successive overrelaxation (SSOR), block incomplete Cholesky (BIC) factorization or modified incomplete Cholesky (MIC) factorization, are effective. Nevertheless, based on eigendecompositions of the C_ℓ ($\ell = 1, 2, 3$) in (15) with the diagonal matrix B_ε, an efficient preconditioner M of the form

$$M = C^*C - \tau I \tag{31}$$

has been employed in [13], with some constant τ, for solving (30). The associated preconditioned linear system

$$(C^*C - \tau I)\mathbf{z} = \mathbf{d}$$

can be rewritten using eigendecompositions of the matrices C_ℓ ($\ell = 1, 2, 3$) in (15) of the following form:

$$(I_3 \otimes \Lambda_q - \tau I)\,\tilde{\mathbf{z}} = \left(I - \tau^{-1}\begin{bmatrix}\Lambda_1\\ \Lambda_2\\ \Lambda_3\end{bmatrix}\begin{bmatrix}\Lambda_1^* & \Lambda_2^* & \Lambda_3^*\end{bmatrix}\right)(I_3 \otimes T)^*\,\mathbf{d}, \tag{32a}$$

$$\mathbf{z} = (I_3 \otimes T)\,\tilde{\mathbf{z}}. \tag{32b}$$

The solution $\mathbf{z}$ in (32) can be efficiently computed using Algorithms 3 and 4, as shown in Algorithm 8. Moreover, a preconditioner M of the form given in (31) can also be used as a preconditioner for solving the nonlinear eigenvalue problems [15] arising from 3D dispersive PCs.

Algorithm 8 [13] Solving the preconditioned linear system $(C^*C - \tau I)\mathbf{z} = \mathbf{d}$.

Input: Lattice vectors $\mathbf{a}_1$, $\mathbf{a}_2$, $\mathbf{a}_3$, grid numbers n_1, n_2, n_3, constant τ and the right hand side vector $\mathbf{d}$.
Output: the solution $\mathbf{z}$.

1: Call Algorithm 2 to compute Λ_1, Λ_2, Λ_3, Λ_q, $D_{\mathbf{a}_1}$, $G_{\mathbf{y}}$, $D_{\hat{\mathbf{a}}_2,i}$, and $G_{\mathbf{z},i}$ for $i = 1, \ldots, n_1$.
2: Call Algorithm 4 with $D_{\mathbf{a}_1}$, $D_{\hat{\mathbf{a}}_2,i}$, and $G_{\mathbf{z},i}$ for $i = 1, \ldots, n_1$ to compute $\mathbf{z} = (I_3 \otimes T^*)\mathbf{d}$.
3: Update $\mathbf{z} := \mathbf{z} - \tau^{-1}\begin{bmatrix}\Lambda_1\\ \Lambda_2\\ \Lambda_3\end{bmatrix}\begin{bmatrix}\Lambda_1^* & \Lambda_2^* & \Lambda_3^*\end{bmatrix}\mathbf{z}$.
4: Compute $\tilde{\mathbf{z}} = (I_3 \otimes \Lambda_q - \tau I)^{-1}\mathbf{z}$.
5: Call Algorithm 3 with $D_{\mathbf{a}_1}$, $G_{\mathbf{y}}$, and $G_{\mathbf{z},i}$ for $i = 1, \ldots, n_1$ to compute $\mathbf{z} = (I_3 \otimes T)\tilde{\mathbf{z}}$.

5.3 Null-space free method

In Subsection 5.2, we have proposed an appropriate preconditioner M in (31) for solving linear systems of the form given in (30). However, according to Theorem 2, each GEP of the form given in (26) has n eigenvalues that are equal to zero. This substantial null space poses the main computational challenge faced when we try to find some smallest positive eigenvalues (usually 6 to 10 eigenvalues are needed for computing the corresponding band structure) of (26), as it will seriously affect the convergence of the eigensolver [13, 17]. In addition, it is not straightforward to determine the shift value σ in (34). The null-space free method [11] has been proposed to address this challenge. By using the SVD of C in (18c), the enormous null space can be deflated, and a $3n \times 3n$ GEP of the form given in (26a) and (26b) can be transformed into the following $2n \times 2n$ NFSEP:

$$A_r \mathbf{x} \equiv \left(\Sigma_r V_r^* B_\varepsilon^{-1} V_r \Sigma_r\right) \mathbf{x} = \lambda \mathbf{x}, \tag{33a}$$

with

$$\mathbf{e} = B_\varepsilon^{-1} V_r \Sigma_r \mathbf{x} \quad \text{and} \quad \mathbf{h} = U_r \mathbf{x}. \tag{33b}$$

The transformation from the GEP in (26) to the NFSEP in (33) is derived in [11].

Since the coefficient matrix A_r in (33a) is Hermitian and positive definite, we can effectively find the desired eigenvalues and eigenvectors by applying the standard inverse Lanczos method. In each step of the inverse Lanczos method, it is necessary to solve the linear system

$$\left(\Sigma_r V_r^* B_\varepsilon^{-1} V_r \Sigma_r\right) \mathbf{y} = \mathbf{b},$$

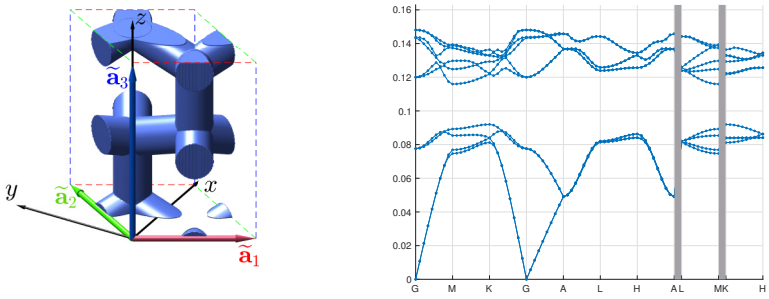
which is equivalent to

$$\left(V_r^* B_\varepsilon^{-1} V_r\right) \tilde{\mathbf{y}} = \Sigma_r^{-1} \mathbf{b}, \quad \mathbf{y} = \Sigma_r^{-1} \tilde{\mathbf{y}}. \tag{34}$$

As shown in [11], the condition number of $V_r^* B_\varepsilon^{-1} V_r$ is small; it depends on only ε and is independent of the matrix dimensions. Therefore, only a few iterations of the conjugate gradient (CG) method without any preconditioner are needed to solve (34), even if the dimension of the matrix exceeds one million. As shown in Algorithm 9, the most expensive operation in the CG method, namely, $V_r^* B_\varepsilon^{-1} V_r \mathbf{v}$, can be efficiently computed by means of Algorithms 3 and 4.

Algorithm 9 [14] Matrix-vector multiplication $\mathbf{u} = V_r^* B_\varepsilon^{-1} V_r \mathbf{v}$.

Input: Φ_1, Φ_2, $D_{\mathbf{a}_1}$, $G_{\mathbf{y}}$, $D_{\hat{\mathbf{a}}_2,i}$, and $G_{\mathbf{z},i}$ for $i = 1, \ldots, n_1$, the diagonal matrix B_ε and the vector $\mathbf{v}$.
Output: the vector $\mathbf{u}$.
1: Compute $\tilde{\mathbf{u}} = [\Phi_1, \Phi_2]\mathbf{v}$.
2: Call Algorithm 3 with $D_{\mathbf{a}_1}$, $G_{\mathbf{y}}$, and $G_{\mathbf{z},i}$ for $i = 1, \ldots, n_1$ to compute $\mathbf{z} = (I_3 \otimes T)\tilde{\mathbf{u}}$.
3: Compute $\tilde{\mathbf{u}} = B_\varepsilon^{-1}\mathbf{z}$.
4: Call Algorithm 4 with $D_{\mathbf{a}_1}$, $D_{\hat{\mathbf{a}}_2,i}$, and $G_{\mathbf{z},i}$ for $i = 1, \ldots, n_1$ to compute $\mathbf{z} = (I_3 \otimes T^*)\tilde{\mathbf{u}}$.
5: Compute $\mathbf{u} = [\Phi_1, \Phi_2]^*\mathbf{z}$.



(a) Wurtzite structure (AB_hP4_186_b_b) in [26]

(b) Band structure

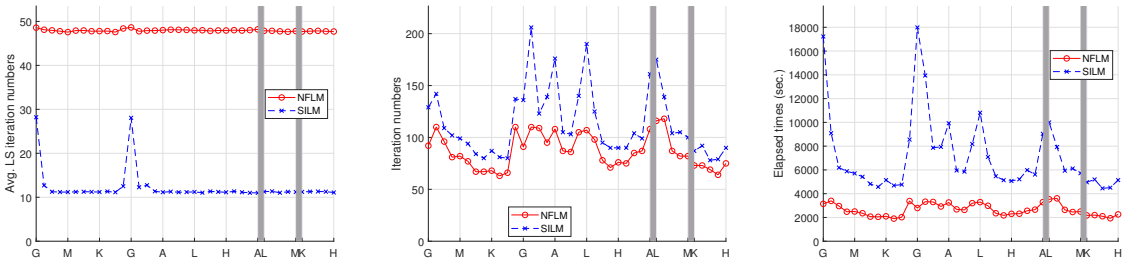
Fig. 5. (a) Schematic of a 3D PC with a hexagonal lattice and (b) the associated band structure.

5.4 Performance of the null-space free method

As a benchmark, we consider the medium structure shown in Figure 5(a), which is a hexagonal lattice consisting of spheres and circular cylinders, both with a radius of 0.6. The coefficient vectors $\{\tilde{\mathbf{s}}_1, \ldots, \tilde{\mathbf{s}}_4\}$ in (29) are taken to be $\tilde{\mathbf{s}}_1 = [\frac{1}{3}, \frac{2}{3}, 0.3748]$, $\tilde{\mathbf{s}}_2 = [\frac{2}{3}, \frac{1}{3}, 0.8748]$, $\tilde{\mathbf{s}}_3 = [\frac{1}{3}, \frac{2}{3}, 0]$, and $\tilde{\mathbf{s}}_4 = [\frac{2}{3}, \frac{1}{3}, \frac{1}{2}]$. The associated lattice translation vectors are $\tilde{\mathbf{a}}_1 = \begin{bmatrix} \frac{a}{2} & -\frac{a\sqrt{3}}{2} & 0 \end{bmatrix}^\top$, $\tilde{\mathbf{a}}_2 = \begin{bmatrix} \frac{a}{2} & \frac{a\sqrt{3}}{2} & 0 \end{bmatrix}^\top$, and $\tilde{\mathbf{a}}_3 = \begin{bmatrix} 0 & 0 & c \end{bmatrix}^\top$, with $a = 3.8227$ and $c = 6.2424$. We take permittivities of $\varepsilon_{in} = 13$ and $\varepsilon_o = 1$ and show the corresponding band structure in Figure 5(b).

The MATLAB function `eigs`, implemented with the shift-and-invert Lanczos method (SILM) and a stopping criterion of 10^{-12}, is used to solve the GEP in (26a), and `bicgstabl` with a preconditioner M (31) is used to address the associated linear system (30a). Here, the shift value σ is taken to be 0.01. On the other hand, `eigs` with a stopping criterion of 10^{-12}, referred to as the null-space free Lanczos method (NFLM), is used to solve the NFSEP in (33), and `pcg` without preconditioning is used to solve (34). We take $n_1 = n_2 = n_3 = 120$; accordingly, the dimensions of the coefficient matrices in (26a) and (33) are $3n_1^3 = 5,184,000$ and $2n_1^3 = 3,456,000$, respectively.

Figure 6(a) illustrates the average numbers of iterations required for solving (30a) and (34), which are fewer than 30 and 50, respectively. These results indicate that the preconditioner in the SILM is quite adequate and that the coefficient matrix in the linear system given in (34) is well-conditioned. Figures 6(b) and 6(c) show the numbers of iterations and the elapsed CPU times for solving the eigenvalue problems corresponding to each of the wave vectors using the SILM and NFLM. Figure 6(b) shows that the NFLM requires fewer iterations than the SILM. This observation indicates that the enormous null space affects the convergence behavior of the SILM, while the zero eigenvalues are deflated in the NFLM. Consequently, as shown in Figure 6(c), the NFLM outperforms the SILM in terms of computation time for all benchmark wave vectors $\mathbf{k}$, even when an effective preconditioner is provided for the SILM.



(a) Average numbers of iterations (b) Numbers of iterations of the SILM and NFLM (c) CPU times of the SILM and NFLM

Fig. 6. (a) Average numbers of iterations of `bicgstabl` and `pcg`, (b) numbers of iterations of the SILM and NFLM, and (c) CPU times of the SILM and NFLM.

Note that the SILM requires a predetermined shift value σ. However, it is not easy to choose a suitable shift value a priori to ensure a suitable overall computation efficiency. The NFLM does not have this drawback because the coefficient matrix A_r in the NFSEP given in (33) is Hermitian and positive definite. Based on this property, we have developed an efficient package, FAME, for solving GEPs of the form given in (26) using the null-space free method.

6 FAME PACKAGE

In this section, we introduce our proposed package "FAME" for simulating the band structures of 3D PCs with all 14 Bravais lattices using MATLAB or an NVIDIA GPU. The source code can be downloaded from the following website:

https://sites.google.com/g2.nctu.edu.tw/fame

To simulate the band structure of a 3D PC, we need to solve a sequence of GEPs of the form given in (26) with various wave vectors $\mathbf{k}$. For each GEP, some of the smallest positive eigenvalues are of interest. In Section 5.3, we have introduced the efficient null-space free method for solving (26). FAME is a package designed for the implementation of this null-space free method in MATLAB or CUDA code; the corresponding implementations of the package are called FAME$_{\mathrm{m}}$ and FAME$_{\mathrm{g}}$, respectively. The FAME package integrates Algorithms 2, 3, 4, and 9 into the inverse Lanczos method and the CG method without any preconditioner (see Step 6 of Algorithm 10) to compute a few of the smallest positive eigenvalues of (33). We summarize all of the processes executed by FAME in Algorithm 10.

We have established models of all 14 Bravais lattices, as listed in Section 1, and the associated first Brillouin zones in the proposed FAME package. Several crystal structures have been implemented as benchmark problems. Users can choose such benchmark problems with given input parameters, such as the permittivities or the radii of the spheres and cylinders, to simulate the associated band structures. They can also provide isosurfaces of their own to create the corresponding domain $\Omega_{p,m}$ using Algorithm 2 and integrate it into FAME to compute the related band structure.

We have implemented FAME$_{\mathrm{m}}$ as a MATLAB app. In this app, users can choose any built-in benchmark problem and visualize the corresponding crystal structure instantly. As an alternative, we have also

Algorithm 10 FAME

Input: Lattice translation vectors $\tilde{\mathbf{a}}_1$, $\tilde{\mathbf{a}}_2$, $\tilde{\mathbf{a}}_3$, electric and vacuum permittivities ε_{in} and ε_o, grid numbers n_1, n_2, n_3, and wave vector $\mathbf{k}$.

Output: target eigenpairs $\{(\lambda_i, \mathbf{e}_i)\}_{i=1}^{p}$.

1: Use Algorithm 1 to compute $\mathbf{a}_1$, $\mathbf{a}_2$, $\mathbf{a}_3$, and the permutation matrix Π.
2: Define $\Omega_{p,m}$ from the given isosurface or by using Algorithm 7 with giving $\{\tilde{\mathbf{a}}_1, \tilde{\mathbf{a}}_2, \tilde{\mathbf{a}}_3\}$, $\{\tilde{\mathbf{s}}_j\}_{j=1}^{s}$ and $\{\tilde{\mathbf{c}}_{t,k}, \tilde{\mathbf{c}}_{b,k}\}_{k=1}^{t}$.
3: Use Algorithm 6 with $\{\tilde{\mathbf{a}}_1, \tilde{\mathbf{a}}_2, \tilde{\mathbf{a}}_3\}$, $\{\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3\}$, $\{\varepsilon_{in}, \varepsilon_o\}$, $\{n_1, n_2, n_3\}$, $\Omega_{p,m}$ and Π to generate the diagonal matrix B_ε.
4: Use Algorithm 2 with $\{\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3\}$, $\{n_1, n_2, n_3\}$ and $\mathbf{k}$ to generate Σ_r, Φ_1, Φ_2, $D_{\mathbf{a}_1}$, $D_{\hat{\mathbf{a}}_2,i}$, and $D_{\hat{\mathbf{a}}_3,i,j}$ for $i = 1, \ldots, n_1$ and $j = 1, \ldots, n_2$.
5: Compute $G_{\mathbf{y}}$ and $G_{\mathbf{z},i}$ according to (19) for $i = 1, \ldots, n_1$.
6: Use inverse Lanczos method to compute the target eigenpairs $\{(\lambda_i, \mathbf{x}_i)\}_{i=1}^{p}$ of the NFSEP (33). In each iteration, use the CG method with the matrix-vector multiplication in Algorithm 9 by using B_ε, Φ_1, Φ_2, $D_{\mathbf{a}_1}$, $G_{\mathbf{y}}$, and $D_{\hat{\mathbf{a}}_2,i}$, $G_{\mathbf{z},i}$ for $i = 1, \ldots, n_1$ to solve (34).
7: **for** $i = 1, \ldots, p$ **do**
8: Compute $\tilde{\mathbf{u}} = [\Phi_1, \Phi_2]\Sigma_r \mathbf{x}_i$.
9: Call Algorithm 3 with $D_{\mathbf{a}_1}$, $G_{\mathbf{y}}$, and $G_{\mathbf{z},i}$ for $i = 1, \ldots, n_1$ to compute $\mathbf{z} = (I_3 \otimes T)\tilde{\mathbf{u}}$.
10: Compute $\mathbf{e}_i = B_\varepsilon^{-1}\mathbf{z}$.
11: **end for**

implemented our null-space free method in the form of MATLAB functions. Users can use these functions to compute target eigenpairs in a given background environment. In addition, the matrices C and B_ε can be saved to a .mat file if needed.

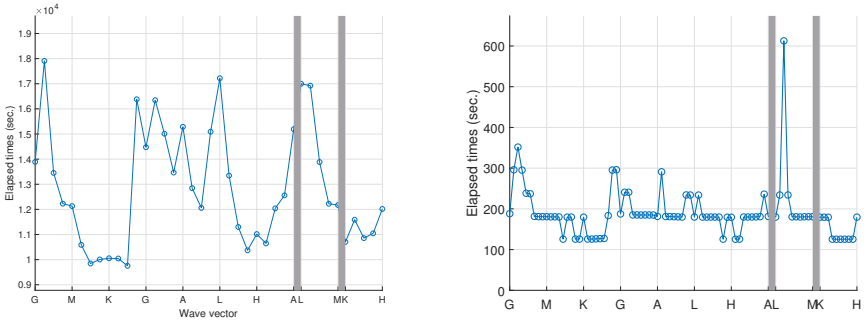
In FAMEg, we have implemented Steps 1 – 3 on a CPU architecture and the other steps on a GPU architecture to reduce the required communication between the CPU and GPU. The Lanczos method and the CG method are implemented as CUDA codes to solve the NFSEP (33) and the associated linear system (34). The high efficiency of FAMEg with a matrix of dimension of 3.5 million for a face-centered cubic lattice with a diamond structure [3, 11], a body-centered cubic lattice with a gyroid structures [24, 33], an orthorhombic lattice with an inverse woodpile structure [36, 37], and a hexagonal lattice consisting of hollow cylinders connected by micropillars [35] has been presented in [16]. In the following, we demonstrate the performance of FAMEg for various other lattices with material structures described in [26].

6.1 Benchmark problems

In this subsection, we present numerical experiments performed using FAMEg on the NVIDIA Tesla P100 GPU. Subject to the GPU memory limitations, we chose the dimension of the coefficient matrix A_r in (33) to be as large as possible. We first demonstrate the efficiency of FAME$_{\mathrm{m}}$ and FAMEg for solving the NFSEP in (33) for a hexagonal lattice, as shown in Figure 5(a). Here, we take $(n_1, n_2, n_3) = (256, 157, 136)$ and $(276, 169, 146)$ for FAME$_{\mathrm{m}}$ and FAMEg, respectively. The dimensions of A_r are $10,932,224$ in the first case and $13,620,048$ in the second. The associated timing performance is shown in Figure 7.

Next, we consider three other benchmark lattices with material structures as proposed in [26] and permittivities of $\varepsilon_{in} = 13$ and $\varepsilon_o = 1$ to further demonstrate the performance of FAMEg.

(1) Body-centered orthorhombic lattice with $a = 3.1440$, $b = 3.1280$, and $c = 7.6770$, as shown in Figure 8(a). The AFLOW prototype label in [26] is AB2_oI6_71_a_i. The radii of both are spheres



(a) $\mathrm{FAME_m}$ (b) $\mathrm{FAME_g}$

Fig. 7. Timing performance for $\mathsf{FAME_m}$ and $\mathsf{FAME_g}$.

and circular cylinders are equal to 0.4. The coefficient vectors $\{\tilde{\mathbf{s}}_j\}_{j=1}^3$ in (29) are $\tilde{\mathbf{s}}_1 = [0,0,0]$, $\tilde{\mathbf{s}}_2 = [0.3390, 0.3390, 0]$ and $\tilde{\mathbf{s}}_3 = [0.6610, 0.6610, 0]$. n_1, n_2 and n_3 are taken to be 248, 215, and 124, respectively, meaning that the corresponding dimension of A_r is $13,223,360$.

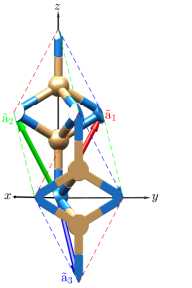
(2) Simple tetragonal lattice with $a = 4.9570$ and $c = 6.8903$ as shown in Figure 8(c). The AFLOW prototype label is A2B_tP12_92_b_a. The radii of both the spheres and circular cylinders are equal to 1.0. n_1, n_2, and n_3 are taken to be 168, 168, and 234, respectively, meaning that the corresponding dimension of A_r is $13,208,832$.

(3) Rhombohedral lattice with $a = 3.7595$ and $\alpha = 1.4137$, as shown in Figure 8(e). The AFLOW prototype label is A_hR2_166_c. The radii of both the spheres and circular cylinders are equal to 0.75. The coefficient vectors $\{\tilde{\mathbf{s}}_j\}_{j=1}^2$ in (29) are $\tilde{\mathbf{s}}_1 = [0.2275, 0.2275, 0.2275]$ and $\tilde{\mathbf{s}}_2 = [0.7725, 0.7725, 0.7725]$. n_1, n_2, and n_3 are taken to be 192, 190, and 188, respectively, meaning that the corresponding dimension of A_r is $13,716,480$.

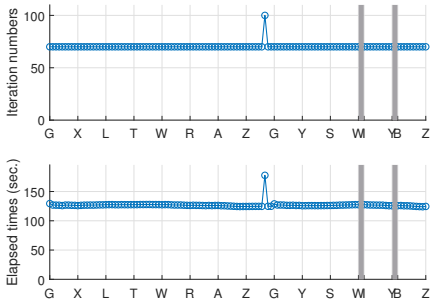
The stopping criteria for both the inverse Lanczos method and the CG method were set to 10^{-12}, and the ten smallest positive eigenvalues were computed. The resulting timing performance for the three benchmark PCs listed above is shown in Figures 8(b), 8(d), and 8(f), respectively. The average computation times in Figures 7(a), 7(b), 8(b), 8(d), and 8(f) are 12897, 190.5, 126.8, 130.4 and 133.1 seconds, respectively. These results show that for eigenvalue problems with the dimension of the coefficient matrices exceeding 13 million, the proposed FAME package is a fast eigensolver for 3D PCs. This efficiency is attributed to the null-space free method and the ability to solve the well-conditioned linear system in (34) using fast FFT-based matrix-vector multiplications as shown in Algorithms 3 and 4. Here, fewer than 45 iterations of the CG method are necessary for all of these benchmark PCs.

6.2 Performance analysis

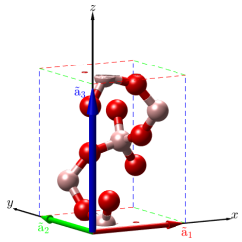
We now present a performance analysis of FAMEg based on the NVIDIA Visual Profiler. The results are shown in Figure 9. In [16], an efficient scheme was developed to reduce the number of launches of the GPU kernel. We combined Step 3 with Step 4 (for computing $\widetilde{Q}_{\mathbf{y}}$) and Step 6 with Step 7 (for computing $\widetilde{Q}_{\mathbf{x}}$) in Algorithm 3 and combined Step 4 with Step 5 (for computing $\widetilde{P}_{e\mathbf{y}}$) in Algorithm 4 to form a single GPU



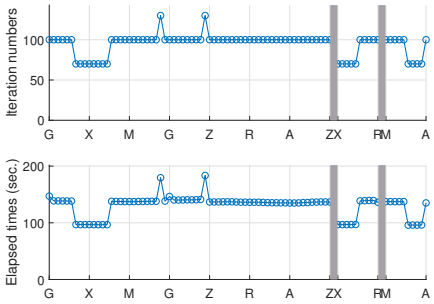
(a) Body-centered orthorhombic lattice



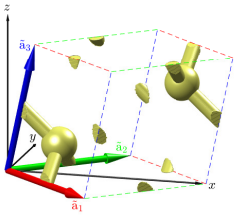
(b) Numbers of iterations and timing performance



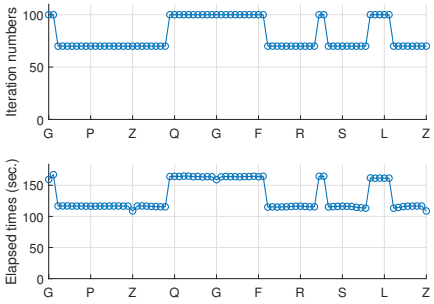
(c) Simple tetragonal lattice



(d) Numbers of iterations and timing performance



(e) Rhombohedral lattice



(f) Numbers of iterations and timing performance

Fig. 8. Schematic of 3D PCs (left) and the corresponding numbers of iterations of the Lanczos method and the associated timing performance (right).

kernel in each case. Under this scheme, the amounts of time consumed to compute $\widetilde{Q}_{\mathbf{x}}$, $\widetilde{Q}_{\mathbf{y}}$, vec$(D_{\mathbf{a}_1}\widetilde{Q}_{u\mathbf{x}})$, $\widetilde{P}_{e\mathbf{x}}$, $\widetilde{P}_{e\mathbf{y}}$, and $\widetilde{P}_{e\mathbf{z}}$ account for approximately 5.4%, 5.3%, 5.3%, 5.3%, 6.5%, and 6.3%, respectively, of the total time. The percentages for $\widetilde{P}_{e\mathbf{y}}$ and $\widetilde{P}_{e\mathbf{z}}$ are more significant than the others, resulting in greater efficiency in computing $T\mathbf{q}$ than in computing $T^*\mathbf{p}$, as shown in Figures 2(b) and 2(c). Moreover, the time needed for matrix-vector multiplications with the matrices V_r and V_r^* when solving the linear system in

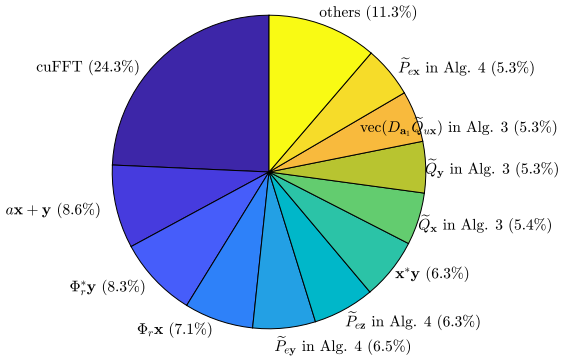


Fig. 9. Percentages of computation time consumed to solve the eigenvalue problem for each kernel.

(34) is approximately 73.8% of the total time, while the time needed to call cuFFT for the 1D backward and forward FFTs is only 24.3% of the total. The matrix-vector multiplications with the matrices V_r and V_r^* are the most computationally expensive steps of solving the NFSEP in (33). The results presented in Figure 2(a) demonstrate that these two matrix-vector multiplications can be performed highly effectively. Efficient matrix-vector multiplication is one of the crucial advantages of our developed FAME package. The other critical advantage of FAME is that the numbers of necessary iterations of the Lanczos method and the CG method are small, as shown in Figures 6 and 8, because of the deflation of the substantial null space.

7 CONCLUSION

In this article, we numerically determine the band structure of three-dimensional photonic crystals by solving a series of large-scale GEPs. Solving such GEPs is a computational challenge. To this end, we propose a package called "FAME" to efficiently address the associated GEPs arising from Yee's discretization. FAME combines the null-space free method with FFT-based matrix-vector multiplications. The null-space free method deflates the null space corresponding to the zero eigenvalues of the eigenvalue problem and improves the convergence of the Lanczos method. The FFT-based matrix-vector multiplications significantly reduce the computational cost of the conjugate gradient method for solving a linear system with a well-conditioned coefficient matrix. These advantages allow the proposed package to efficiently simulate the band structure of a 3D photonic crystal even when the matrix dimensionality exceeds 13 million.

ACKNOWLEDGMENTS

T.-M. Huang was partially supported by the Ministry of Science and Technology of Taiwan (MoST) 108-2115-M-003-012-MY2. T. Li was supported in part by the National Natural Science Foundation of China (NSFC) 11971105. W.-W. Lin was partially supported by MoST 106-2628-M-009-004-. This work was also partially supported by the National Centre of Theoretical Sciences (NCTS) and ST Yau Centre in Taiwan, the Shing-Tung Yau Center and the Big Data Computing Center of Southeast University. The

numerical calculation of this work was partially performed on TianHe-2, thanks to the support of the National Supercomputing Center in Guangzhou (NSCC-GZ).

APPENDIX A

(i) for $\mathbf{a}_3(2) \geq 0$ and $m_4 \equiv m_2 - m_1 \geq 0$,

$$J_3 = \begin{bmatrix} 0 & e^{-\imath 2\pi \mathbf{k}\cdot(\mathbf{a}_2+(\rho_1-\rho_2)\mathbf{a}_1)} I_{m_3} \otimes J_{2,m_4} \\ e^{\imath 2\pi \rho_2 \mathbf{k}\cdot\mathbf{a}_1} I_{n_2-m_3} \otimes J_{2,m_2} & 0 \end{bmatrix}.$$

(ii) for $\mathbf{a}_3(2) \geq 0$ and $m_4 < 0$,

$$J_3 = \begin{bmatrix} 0 & e^{-\imath 2\pi \mathbf{k}\cdot(\mathbf{a}_2+(\rho_1-\rho_2-1)\mathbf{a}_1)} I_{m_3} \otimes J_{2,n_1+m_4} \\ e^{\imath 2\pi \rho_2 \mathbf{k}\cdot\mathbf{a}_1} I_{n_2-m_3} \otimes J_{2,m_2} & 0 \end{bmatrix}.$$

(iii) for $\mathbf{a}_3(2) < 0$ and $m_5 \equiv m_1 + m_2 \leq n_1$,

$$J_3 = \begin{bmatrix} 0 & e^{\imath 2\pi \rho_2 \mathbf{k}\cdot\mathbf{a}_1} I_{m_3} \otimes J_{2,m_2} \\ e^{\imath 2\pi \mathbf{k}\cdot(\mathbf{a}_2+(\rho_2+\rho_1)\mathbf{a}_1)} I_{n_2-m_3} \otimes J_{2,m_5} & 0 \end{bmatrix}.$$

(iv) for $\mathbf{a}_3(2) < 0$ and $m_5 > n_1$,

$$J_3 = \begin{bmatrix} 0 & e^{\imath 2\pi \rho_2 \mathbf{k}\cdot\mathbf{a}_1} I_{m_3} \otimes J_{2,m_2} \\ e^{\imath 2\pi \mathbf{k}\cdot(\mathbf{a}_2+(\rho_2+\rho_1-1)\mathbf{a}_1)} I_{n_2-m_3} \otimes J_{2,m_5-n_1} & 0 \end{bmatrix},$$

where

$$J_{2,m} \equiv \begin{bmatrix} 0 & e^{-\imath 2\pi \mathbf{k}\cdot\mathbf{a}_1} I_m \\ I_{n_1-m} & 0 \end{bmatrix}.$$

APPENDIX B

Here we analyze the complexity of the matrix-vector multiplications $T\mathbf{q}$ and $T^*\mathbf{p}$ processed in Algorithm 3 and 4, respectively.

- In Algorithm 3, the complexities without parallelization in Line 2-3, Line 5-6, and Line 7-8 are

$$\begin{aligned} c_{11} &\equiv (n_1 n_2) \times \mathcal{O}(n_3 \log n_3) + n_1 \times (n_2 n_3), \\ c_{12} &\equiv (n_1 n_3) \times \mathcal{O}(n_2 \log n_2) + n_3 \times (n_2 n_1), \\ c_{13} &\equiv (n_2 n_3) \times \mathcal{O}(n_1 \log n_1) + (n_2 n_3) \times n_1 + n, \end{aligned}$$

  respectively.
- In Algorithm 4, the complexities without parallelization in Line 2-3, Line 5-6, and Line 7-8 are

$$\begin{aligned} c_{21} &\equiv n_3 \times (n_2 \times n_1) + (n_2 n_3) \times \mathcal{O}(n_1 \log n_1), \\ c_{22} &\equiv n_1 \times (n_3 \times n_2) + (n_1 n_3) \times \mathcal{O}(n_2 \log n_2), \\ c_{23} &\equiv n_1 \times (n_2 n_3) + (n_1 n_2) \times \mathcal{O}(n_3 \log n_3) + n, \end{aligned}$$

  respectively.

Therefore, the complexities for computing $T\mathbf{q}$ and $T^*\mathbf{p}$ are both equal to

$$c = 4n + (n_1 n_2 \mathcal{O}(n_3 \log n_3) + n_1 n_3 \mathcal{O}(n_2 \log n_2) + n_2 n_3 \mathcal{O}(n_1 \log n_1)).$$

REFERENCES

[1] D. Boffi, M. Conforti, and L. Gastaldi. 2006. Modified edge finite elements for photonic crystals. *Numer. Math.* 105 (September 2006), 249–266.

[2] Z. Chen, Q. Du, and J. Zou. 2000. Finite element methods with matching and nonmatching meshes for Maxwell equations with discontinuous coefficients. *SIAM J. Numer. Anal.* 37 (July 2000), 1542–1570.

[3] R.-L. Chern, C.-Chung Chang, Chien-C. Chang, and R.-R. Hwang. 2004. Numerical study of three-dimensional photonic crystals with large band gaps. *J. Phys. Soc. Japan* 73 (November 2004), 727–737.

[4] R.-L. Chern, H.-E. Hsieh, T.-M. Huang, W.-W. Lin, and W. Wang. 2015. Singular Value Decompositions for Single-Curl Operators in Three-Dimensional Maxwell's Equations for Complex Media. *SIAM J. Matrix Anal. Appl.* 36 (March 2015), 203–224.

[5] D. C. Dobson and J. Pasciak. 2001. Analysis for an algorithm for computing electromagnetic Bloch modes using Nedelec spaces. *Comp. Meth. Appl. Math.* 1 (June 2001), 138–153.

[6] W. Dörfler, A. Lechleiter, M. Plum, G. Schneider, and C. Wieners. 2011. *Photonic Crystals: Mathematical Analysis and Numerical Approximation*. Springer Basel AG.

[7] C. Engström, C. Hafner, and K. Schmidt. 2009. Computations of lossy Bloch waves in two-dimensional photonic crystals. *J. Comput. Theor. Nanosci.* 6 (March 2009), 1–9.

[8] C. Engström and M. Wang. 2010. Complex dispersion relation calculations with the symmetric interior penalty method. *Int. J. Numer. Meth. Engng.* 84 (October 2010), 849–863.

[9] P. G. Etchegoin, E. C. Le Ru, and M. Meyer. 2006. An analytic model for the optical properties of gold. *J. Chem. Phys.* 125 (October 2006), 164705.

[10] S. Guo, F. Wu, S. Albin, and R. S Rogowski. 2004. Photonic band gap analysis using finite-difference frequency-domain method. *Opt. Express* 12 (April 2004), 1741–1746.

[11] T.-M. Huang, H.-E. Hsieh, W.-W. Lin, and W. Wang. 2013. Eigendecomposition of the discrete double-curl operator with application to fast eigensolver for three dimensional photonic crystals. *SIAM J. Matrix Anal. Appl.* 34 (April 2013), 369–391.

[12] T.-M. Huang, H.-E. Hsieh, W.-W. Lin, and W. Wang. 2013. Matrix representation of the double-curl operator for simulating three dimensional photonic crystals. *Math. Comput. Model.* 58 (July 2013), 379–392.

[13] T.-M. Huang, H.-E. Hsieh, W.-W. Lin, and W. Wang. 2014. Eigenvalue solvers for three dimensional photonic crystals with face-centered cubic lattice. *J. Comput. Appl. Math.* 272 (December 2014), 350–361.

[14] T.-M. Huang, T. Li, W.-D. Li, J.-W. Lin, W.-W. Lin, and H. Tian. 2018. *Solving Three Dimensional Maxwell Eigenvalue Problem with Fourteen Bravais Lattices*. Technical Report. arXiv:1806.10782.

[15] T.-M. Huang, W.-W. Lin, and V. Mehrmann. 2016. A Newton-type method with nonequivalence deflation for nonlinear eigenvalue problems arising in photonic crystal modeling. *SIAM J. Sci. Comput.* 38 (March 2016), B191–B218.

[16] T.-M. Huang, W.-W. Lin, H. Tsai, and W. Wang. 2019. Highly efficient GPU eigensolver for three-dimensional photonic crystal band structures with any Bravais lattice. *Comput. Phys. Commun.* 245 (December 2019), 106841.

[17] Y.-L. Huang, T.-M. Huang, W.-W. Lin, and W.-C. Wang. 2015. A null space free Jacobi-Davidson iteration for Maxwell's operator. *SIAM J. Sci. Comput.* 37, 1 (January 2015), A1–A29.

[18] J. Jin. 2002. *The finite element method in electromagnetics*. John Wiley, New York, NY.

[19] J. D. Joannopoulos, S. G. Johnson, J. N. Winn, and R. D. Meade. 2008. *Photonic Crystals: Molding the Flow of Light*. Princeton University Press, Princeton, NJ.

[20] S. G. Johnson and J. D. Joannopoulos. 2001. Block-iterative frequency-domain methods for Maxwell's equations in a planewave basis. *Opt. Express* 8, 3 (January 2001), 173–190.

[21] C. Kittel. 2005. *Introduction to solid state physics*. Wiley, New York, NY.

[22] N. Liu, G. Cai, C. Zhu, Y. Huang, and Q. H. Liu. 2016. The Mixed Finite-Element Method With Mass Lumping for Computing Optical Waveguide Modes. *IEEE J. Sel. Top. Quantum Electron.* 22 (April 2016), 4400709.

[23] N. Liu, L. E. Tobón, Y. Zhao, Y. Tang, and Q. H. Liu. 2015. Mixed Spectral-Element Method for 3-D Maxwell's Eigenvalue Problem. *IEEE Trans. Microwave Theo. Tech.* 63 (February 2015), 317–325.

[24] L.-Z. Lu, L. Fu, J. D Joannopoulos, and M. Soljačić. 2013. Weyl points and line nodes in gyroid photonic crystals. *Nat. Photonics* 7 (March 2013), 294–299.

[25] M. Luo and Q. H. Liu. 2010. Three-dimensional dispersive metallic photonic crystals with a bandgap and a high cutoff frequency. *J. Opti. Soc. Amer. A* 27, 8 (August 2010), 1878–1884.

[26] M. J. Mehl, D. Hicks, C. Toher, O. Levy, R. M. Hanson, G. L. W. Hart, and S. Curtarolo. 2017. The AFLOW Library of Crystallographic Prototypes: Part 1. *Comput. Mater. Sci.* 136 (August 2017), S1–S828.

[27] E. Moreno, D. Erni, and C. Hafner. 2002. Band structure computations of metallic photonic crystals with the multiple multipole method. *Phys. Rev. B* 65 (April 2002), 155120.

[28] G. Mur and A. de Hoop. 1985. A finite-element method for computing three-dimensional electromagnetic fields in inhomogeneous media. *IEEE Trans. Magnetics* 21 (November 1985), 2188–2191.

[29] J.-C. Nédélec. 1986. A new class of mixed finite elements in $\mathbb{R}^3$. *Numer. Math.* 50 (January 1986), 57–81.

[30] J. B. Pendry, A. J. Holden, D. J. Robbins, and W. J. Stewart. 1999. Magnetism from conductors and enhanced nonlinear phenomena. *IEEE Trans. Microwave Theo. Tech.* 47 (December 1999), 2075–2084.

[31] J. B. Pendry, A. J. Holden, W. J. Stewart, and I. Youngs. 1996. Extremely low frequency plasmons in metallic mesostructures. *Phys. Rev. Lett.* 76 (June 1996), 4773–4776.

[32] M. Reed and B. Simon. 1978. Methods of Modern Mathematical Physics. In *Analysis of Operators IV*. Academic Press, San Diego, CA.

[33] A. Schoen. 1970. *Infinite Periodic Minimal Surfaces without Self-intersections*. Technical Report. NASA Technical Note No. D-5541.

[34] A. Vial. 2007. Implementation of the critical points model in the recursive convolution method for modelling dispersive media with the finite-difference time domain method. *J. Opt. A: Pure Appl. Opt.* 9 (July 2007), 745–748.

[35] H. Wang, Lin Xu, H. Y. Chen, and J.-H. Jiang. 2016. Three-dimensional photonic Dirac points stabilized by point group symmetry. *Phys. Rev. B* 93 (June 2016), 235155.

[36] L. A. Woldering, A. P. Mosk, R. W. Tjerkstra, and W. L. Vos. 2009. The influence of fabrication deviations on the photonic band gap of three-dimensional inverse woodpile nanostructures. *J. Appl. Phys.* 105 (May 2009), 093108.

[37] L. A. Woldering, A. P. Mosk, and W. L. Vos. 2014. Design of a three-dimensional photonic band gap cavity in a diamondlike inverse woodpile photonic crystal. *Phys. Rev. B* 90 (May 2014), 115140.

[38] F. Xu, Y. Zhang, W. Hong, K. Wu, and T.-J. Cui. 2003. Finite-difference frequency-domain algorithm for modeling guided-wave properties of substrate integrated waveguide. *IEEE Trans. Microwave Theo. Tech.* 51, 11 (November 2003), 2221–2227.

[39] K. Yee. 1966. Numerical solution of initial boundary value problems involving Maxwell's equations in isotropic media. *IEEE Trans. Antennas and Propagation* 14 (May 1966), 302–307.

[40] C.-P. Yu and H.-C. Chang. 2004. Compact finite-difference frequency-domain method for the analysis of two-dimensional photonic crystals. *Opt. Express* 12, 7 (April 2004), 1397–1408.