scientific reports

Check for updates

OPEN

ZenBand: a numerical solver of photonic crystals with a graphical user interface

Andrius Zinkevičius✉, Ignas Lukošiūnas & Darius Gailevičius

We developed an open-source Plane Wave Expansion Method solver using Python and a custom Tkinter library to solve a design oriented problem of photonic crystal dispersion for known classical examples, custom geometries, and symmetries. Such structures are capable of light confinement, omnidirectional reflection, beam collimation and negative refraction. We dive deeper into the diagonally anisotropic photonic crystals, whose Plane Wave Expansion algorithm is directly embedded in the application. The user interface is present in the developer's repository link: https://github.com/ZenTunturi/ZenBand.

The necessity of open-source numerical software in modern times is of significant importance. Especially open-access solvers that aid in solving differential equations numerically for conservative and non-conservative optical systems[1–3], as well as for data analysis[4–6] and image processing[7–9]. The main difficulty with preexisting commercial software, such as Lumerical, is the high license pricing, data analysis limitations, numerical problem-solving specialization, and extensive tutorial requirements due to the software's complexity. Such limitations, with respect to the developers of the commercial software, delay timely progress in photonic system simulations and system modeling for system designers and engineers.

Nowadays, there are free-to-use alternatives for many widely studied algorithms that have many built-in optimization tools. For example, Meep[10] is a C++ library that simplifies Finite Difference Time Domain (FDTD) simulations by providing simple and intuitive functions. These functions enable users to create a simulation window with only a fraction of the lines of code. Also, the Rigorous Coupled Wave Analysis (RCWA) algorithm is implemented in TORCWA[11] as well as MAXIM[12] software for periodically distributed linear optical materials. The former is a GPU-accelerated Python library with integrated geometric parameter optimization, utilizing Python libraries for automatic differentiation when calculating the gradient of the evaluation function. MAXIM is more suited for users who are not as familiar with programming, as it features a highly intuitive graphical user interface (GUI).

If a numerical model of a counter-top optical system is needed, the FDTD method can appear to be too tedious. A great alternative for this need is presented by "Diffractio"[13]—an open-source Python package that simplifies the analysis of diffraction and interference, both within the scalar and vector optics approaches. This library is intuitively oriented around three modules: sources, masks and fields.

Another field of photonics that numerical software has helped to improve is the fabrication of diffractive optical elements (DOEs). MetaOptics[14] and GDOESII[15] are such applications with simple GUIs, that help users create metasurface phase-masks using simple meta-atom geometries.

Numerous numerical software packages have been developed for various specialized fields of optics. BMP-Matlab[16] is an advanced, computationally efficient algorithm that provides multi-mode analysis through a wide variety of fiber geometries and even accounts for the effect of bends. In addition, PyWolf[17] is a software that computes the propagation of partially coherent light by examining the evolution of a cross-spectral density function of two-dimensional light sources in both near-field and far-field approximations. The software exhibits excellent performance, as the 4D cross-spectral density array elements can be computed in parallel. PtyLab[18] is an open-source ptychographic reconstruction software, that was made with accessibility in mind. The software is multi-lingual—it is available in MATLAB, Python, and Julia.

The progress of CPU-based open-source libraries (such as numpy, scipy, matplotlib, pyvista, and others) shows excellent results in constructing eigenvalue-based algorithm libraries that are applied for electromagnetic simulations[19–22]. Concurrently, suppose the array dimensions are large enough. In that case, various solver libraries are well-written and developed for open-source usage, such as 3D RCWA-based Torcwa, which utilizes PyTorch as its primary library for numerical calculations and analysis of periodic structures. Recent

Laser Research Center, Faculty of Physics, Vilnius University, Saulėtekio Ave. 10, Vilnius, Lithuania. ✉email: andrius.zinkevicius@ff.vu.lt

advancements are being made for photonic crystal slabs, which involve the use of libraries for dispersion calculation for 2D photonic crystals[23–26].

However, with the intense and successful progress being made in open-source library additions, there is a relatively small amount of open software that utilizes these libraries for User Interface (UI) construction. In fact, open-source libraries can be combined with one another to form an open-source software, which in turn provides simplicity in the use of libraries for numerical modeling. Specifically, using PyQt or Tkinter for UI construction. In this work, we provide our own version of a Tkinter-based application with an embedded 2D Plane Wave Expansion Method (2D PWEM) with implemented diagonal anisotropy.

The functions that our application provides include photonic band computations in an oblique grid, the evaluation of iso-frequency contours for spatial dispersion control, and eigen-field computations given a specific symmetry of the PhC distribution. The application offers such functions for the diagonally anisotropic materials. Other graphic options of the User Interface include GIF generation and modern UI designs. The application also includes a function for importing custom-generated permittivity, allowing users to include their own custom permittivity distribution arrays along with the Brillouin zone of interest. Taking these points into account, we provide contributions to the field of computational photonics with our open-source UI:

- 2D PWEM method written in Python using customtkinter application design library, as seen in Fig 1. The greatest contribution presented here is the implemented models for isotropic and diagonally anisotropic materials.
- Cross-validation of already published results that encompass Photonic Crystal structures. Their respective properties, such as self-collimation and negative refractive index, are thoroughly studied.
- Numerical speed comparison study of 2D PWEM in Python and Matlab environments.
- Convergence study of the algorithm for high contrast dielectric Photonic Crystals.

Numerical method

The ZenBand application is based on the Plane Wave Expansion Method, which is convenient to use when dealing with electromagnetic fields in periodic structures[27]. It is easy to implement and is less computationally intensive compared to other methods such as FDFD[28]. The algorithm, that is used in this application, was originally developed by dr. Raymond C. Rumpf[29]. The basic idea relies on Fourier transforming the Maxwell's equations by expanding the electric, magnetic fields, dielectric permittivity and magnetic permeability in a plane wave basis. The set of equations is then rewritten in matrix form and an eigenvalue equation is constructed:

$$\left(\mathrm{K}_x[\![\mu_{r,yy}]\!]^{-1}\mathrm{K}_x + \mathrm{K}_y[\![\mu_{r,xx}]\!]^{-1}\mathrm{K}_y\right) s_z = k_0^2[\![\varepsilon_{r,zz}]\!]s_z, \tag{1}$$

$$\left(\mathrm{K}_x[\![\varepsilon_{r,yy}]\!]^{-1}\mathrm{K}_x + \mathrm{K}_y[\![\varepsilon_{r,xx}]\!]^{-1}\mathrm{K}_y\right) u_z = k_0^2[\![\mu_{r,zz}]\!]u_z; \tag{2}$$

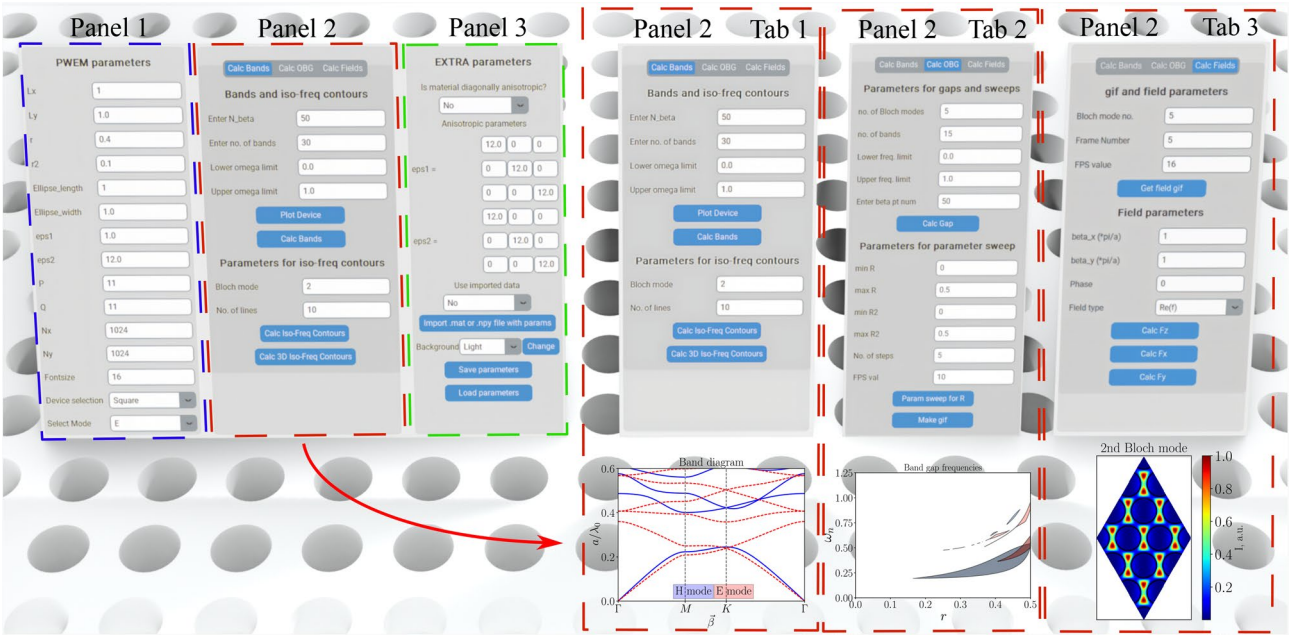

Fig. 1. Application layout scheme for photonic crystal computations. The UI consists of three panels. Panel 1 holds the general parameters, Panel 2 holds the functions and Panel 3 holds additional features. The second panel is made of three tabs - Tab 1 serves band and iso-frequency calculations for generalized symmetries, Tab 2 is for omnidirectional band-gap analysis and Tab 3 is for field calculations.

where K_i is a diagonal wave vector matrix for ith component, s_z is the electric field eigen-vector and u_z is the magnetic field eigen-vector for z component, and double brackets denote convolution matirces. Here $k_0 = \omega/c_0$ is the eigenvalue. The equations become simpler for isotropic materials as permittivity and permeability convolution matrices for every component become the same (given $i = (x, y, z)$): $[\![\varepsilon_{r,ii}]\!] \rightarrow [\![\varepsilon_r]\!]$, $[\![\mu_{r,ii}]\!] \rightarrow [\![\mu_r]\!]$.

A schematic representation of the PWEM formalism is given in Fig. 2a. The blue columns symbolize a dielectric material that stretches to infinity in z direction. The red sinusoidal wave is a representation of a Bloch wave. The big red arrow depicts the Bloch wave vector ($\vec{\beta}$) and the two smaller arrows indicate the polarization convention. The polarization of the field is chosen by the component that oscillates in the *xy* plane. Two types of polarizations are present in the 2D model: Transverse Electric (TE) and Transverse Magnetic (TM) polarizations, which in the model are denoted as *H* and *E* modes, respectively. Such modes are the field components for which the PWEM problem is derived and solved, where the former field is $H_z(x, y)$ and the latter is $E_z(x, y)$.

A simplified 2D PWEM algorithm for the numerical method is presented in Algorithm 1. The process starts with the user defining the unit cell grid. If built-in lattice geometries are used, then direct and reciprocal lattice vectors are defined automatically. Next, convolution matrices are computed and an array of Bloch wave vectors (denoted as $\vec{\beta}$) is constructed. If either band diagrams or iso-frequency contours are calculated, a *for* loop is initialized. For every $\vec{\beta}$ array element, $K_i^{(n)}$ wave vector matrices are constructed (line 2), which are used to formulate the eigenvalue equations (lines 4–5 and 7–8). The equation that is depicted in line number 10 in Algorithm 1 can only solve for z component fields. If either f_x or f_y is required, additional equations have to be solved, which are presented in lines 13–14 and 16–17. The fields are then reshaped into 2D arrays (lines 17–18) and inverse Fourier transformed into their field distributions.

Require: $\varepsilon_r(\vec{r}), \mu_r(\vec{r}), \vec{\mathrm{t}}_{1,2}, \vec{\mathrm{T}}_{1,2}$ ▷ Lattice geometry and grid
Ensure: $[\![\varepsilon_r]\!], [\![\mu_r]\!]$ ▷ Calc convolution matrices
Ensure: $\vec{\beta}_n$ ▷ Init wavevector array
1: **for** n in range $\vec{\beta}$ **do** ▷ Iterate through spatial harmonics
2: $K_x^{(n)}, K_y^{(n)} \rightarrow \vec{\beta}^{(n)} - \vec{\mathrm{T}}$ ▷ Wavevector components
3: **if** mode is 'E' (TM pol.) **then**
4: $\mathrm{A} = K_x [\![\mu_{yy,r}]\!]^{-1} K_x + K_y [\![\mu_{xx,r}]\!]^{-1} K_y$
5: $B = [\![\varepsilon_{zz,r}]\!]$ ▷ Eigenproblem for E mode
6: **else** (TE pol.)
7: $\mathrm{A} = K_x [\![\varepsilon_{r,yy}]\!]^{-1} K_x + K_y [\![\varepsilon_{r,xx}]\!]^{-1} K_y$
8: $B = [\![\mu_{zz,r}]\!]$ ▷ Eigenproblem for H mode
9: **end if**
10: $k_0^2(\beta_x^{(n)}, \beta_y^{(n)}), \mathrm{f_z}(\beta_x^{(n)}, \beta_y^{(n)}) = \mathrm{eigs(A, B)}$ ▷ Eigen-values and vectors
11: **if** is field **then**
12: **if** Mode is 'E' (TM pol.) **then**
13: $\mathrm{f}_x = -\frac{j}{k_0} [\![\mu_{r,xx}]\!]^{-1} K_y \mathrm{f_z}$ ▷ h_x field
14: $\mathrm{f}_y = +\frac{j}{k_0} [\![\mu_{r,yy}]\!]^{-1} K_x \mathrm{f}_z$ ▷ h_y field
15: **else**
16: $\mathrm{f}_x = -\frac{j}{k_0} [\![\varepsilon_{r,xx}]\!]^{-1} K_y \mathrm{f_z}$ ▷ e_x field
17: $\mathrm{f}_y = +\frac{j}{k_0} [\![\varepsilon_{r,yy}]\!]^{-1} K_x \mathrm{f}_z$ ▷ e_y field
18: **end if**
19: reshape f_i from (PQ,) $\rightarrow$ (P,Q) ▷ $i = (x, y, z)$
20: $\mathrm{f}_i \rightarrow$ zeros(N_x,N_y) ▷ Stack spatial harmonics
21: $\mathrm{F}_i = \mathrm{IFFT_{2D}}(\mathrm{f_i})$ ▷ Inverse Fourier Transform
22: **end if**
23: **end for**

Algorithm 1. General 2D PWEM algorithm for photonic band analysis of diagonal anisotropy.

To provide further clarification , a simplified code example for field distribution calculations is provided in Listing 1. Every other function in ZenBand is implemented equivalently. The code starts by extracting parameters and initializing them. Then, the symmetry is defined by real and reciprocal lattice vectors, as well as an array of Bloch wave ($\vec{\beta}$) vectors, when calculating photonic bands. Afterwards, the device's unit cell grid is created.

After properly defining the Bloch wave vector (i.e., the point in the reciprocal lattice), eigenvalue equations are constructed and solved for the chosen polarization, and eigen-vectors are returned. Since these vectors are in Fourier space, they must be reshaped into a 2D array and then inverse-transformed.

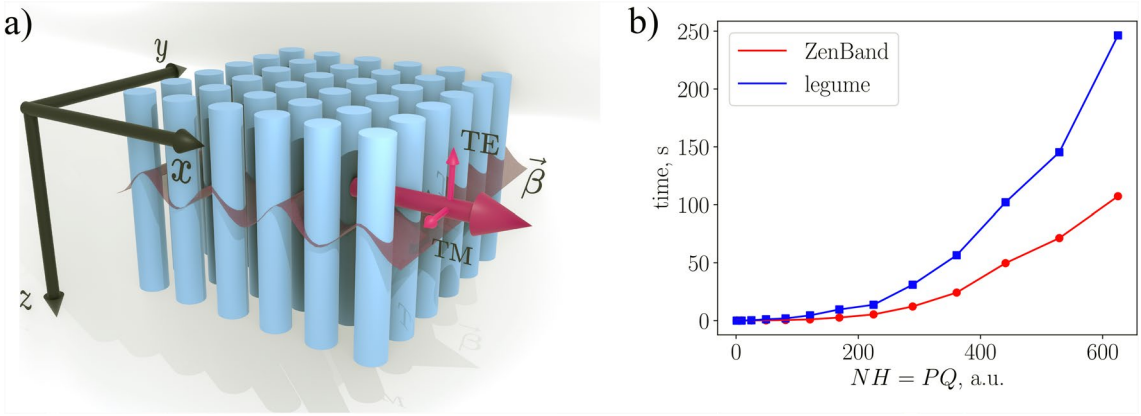

Fig. 2. Simplified depiction of 2D PWEM concept (**a**). This example would be considered as TM polarization or E mode. Bloch wave vector $\vec{\beta}$ indicates the direction of wave propagation in a crystal. Graph (**b**) compares the time it takes to calculate band diagrams for a hexagonal lattice between legume[30] and ZenBand. Letters P and Q indicate the number of spatial harmonics in x and y directions.

```
# Extract parameters from app entries
app.get_params()

# Initialize the parameters and device unit cell grid
params = Params(app)
Device = device()

# define lattice and beta (Bloch wave) vectors
Pwem_params = pwem_params(app, params)
Pwem_params.Symmetry(params);

# calculate device's unit cell grid (square in this example)
Device.Ellipse(params)

# define beta vector
beta_x = app.beta_x_val * np.pi /params.Lx
beta_y = app.beta_y_val * np.pi /params.Ly

# get eigen vectors (V - field vector, m - index for the Bloch mode of interest, WE - k0 values)
if app.sel_mode == 'E':
    V, m, WE = calc_E_mode_field(params.Harmonics[0], params.Harmonics[1], Pwem_params.T1, Pwem_params.
    T2, beta_x, beta_y, Device.ERC, Device.URC, params.norm, app.Bloch_mode_no)
else:
    V, m, WE = calc_E_mode_H_field(params.Harmonics[0], params.Harmonics[1], Pwem_params.T1, Pwem_params
    .T2, beta_x, beta_y, Device.ERC, Device.URC, app.BC, params.norm, app.Bloch_mode_no, app.field_comp)

# Inverse Fourier transform the field
a = field(V, params.Harmonics[0], params.Harmonics[1], m, params.dim[0], params.dim[1], Device.ER,
    params.x, params.y)

# Draw the field
field_fig = single_field(app.background, app.device_selection, Device.ER, app.Bloch_mode_no, params, WE,
     a, beta_x, beta_y, app.phase, app.sel_import, app.sel_field, app.FontSize)
```

Listing 1. 'Calc Fz' function handling of ZenBand.

The performance of ZenBand was tested against legume[30]—a Python package that includes a PWEM solver. Bands for a hexagonal lattice with $r^* = 0.2$ dielectric rods of $\varepsilon_r = 12$ were calculated using different numbers of spatial harmonics from $P = Q = 1$ up to $P = Q = 25$. An equal number of 60 $\vec{\beta}$ points were used in both cases. The time was measured from the point when the physical and numerical parameters are initialized up until the bands are computed. Figure 2b shows that ZenBand diverges more slowly compared to legume, when the number of spatial harmonics increases. A computer with a 12th Gen Intel(R) Core(TM) i3-1215U processor with 16 GB DDR4-3200 RAM was used for the simulations on Python 3.12.7. The maximum CPU frequency of 4.4 GHz could be achieved during runs.

Results

Application layout

The 2D PWEM application consists of three panels, as shown in Fig. 1. The first panel contains entries for PWEM parameters, the second panel features three tabs for executing the calculations, and the third panel provides additional options. PWEM parameters enable the user to conveniently adjust both the 2D device and computational performance. The program offers the freedom to choose between simple yet widely used PhC designs, including cylinders, frames, and rings in a rectangular grid, as well as cylinders in hexagonal and honeycomb grids. In addition, it also allows the user to change the number of periods for the unit cell in x and y directions, the width and eccentricity of the cylinders and the dielectric permittivity values. Computational performance can be adjusted by changing the unit cell grid size and the number of spatial harmonics used in x and y directions.

As mentioned earlier, the second panel is used to initiate the computation and is divided into three tabs. Each command, which is present in the second panel, returns a figure or creates a *.gif* file. If a figure is computed, the program creates a new top-level window containing the *matplotlib* figure and navigation toolbar, which allows the user to make simple manipulations and save the figure in a convenient format.

The first tab contains buttons which initialize band diagram and iso-frequency contour (which can be plotted as both 2D and 3D graphs, the latter of which is shown in Fig. 3b) calculations and hold entries for related parameters such as the number of beta vector points or the number of Bloch modes of interest. This tab also holds the 'Plot Device' button, which returns a figure of the unit cell grid. To give an example, Fig. 3 will be used to display a portion of the functions and effects of different parameters. A honeycomb unit cell with different hole widths is chosen (Fig. 3a) replicating the results of a publication where the topology of 2D photonic crystals was investigated[31]. Lattices with such symmetry, known as valley photonic crystals, are a hot topic at the moment due to their excellent waveguide properties[32–35].

The second tab was made with the omnidirectional bandgap (OBG) analysis in mind. The 'Calc Gap' button returns a band diagram with highlighted bandwidths where OBGs are present. Gaps for E mode (TM pol.) are marked in red, gaps for H mode (TE pol.) are marked in blue, and complete photonic band gaps are colored yellow. This tab also includes commands for parameter sweep—the radius of the structure is increased in a desired interval and the change in OBG placement and bandwidth is plotted. The output of the function is demonstrated in Fig. 3c. The value of r^* was increased from 0 to $0.5a$ while the radius of the other cylinder was kept constant at $r_2^* = 0.4a$. A similar concept is used to create a *.gif* file, which visualizes the change in the radius value within a unit cell and the evolution of the band diagram.

The third tab is dedicated to visualizing Bloch fields in a device. 'Get field gif' command exhibits the evolution of a Bloch mode when the Bloch vector is being dragged across the perimeter of the irreducible Brillouin zone (IBZ). The other three buttons allow the user to visualize the Bloch mode at any point of the reciprocal lattice. When E mode is selected, 'Calc Fz' returns a figure of the electric field z component in the device. 'Calc Fx' and 'Calc Fy' return fields of the magnetic field in the corresponding component. Figure 3d shows the distribution of the real part of the field, while (e) depicts the intensity distribution of the first Bloch mode in TM polarization at the K point of symmetry. Similar results can be found in "Figure 5" of the aforementioned publication[31], where the field leaks deep into the crystal and is concentrated in the dielectric rods.

The third panel is dedicated to extra parameters and options. Since diagonally anisotropic materials can be adapted to the 2D PWEM algorithm, the option to analyze such materials was included. To enhance the functionality of this application, an option has been added to import data. By importing a *.mat* or *.npy* file with the device grid and necessary parameters, a user can compute band diagrams and field distributions for a wide variety of lattices with minimal limitations.

To enhance the user experience, an option to save parameters and enable different background modes was added. Since there are numerous adjustable parameters, saving them eliminates the need to fill them in every time a user wants to use the application, especially when only minor adjustments are desired. The background can also be changed between light and dark modes, providing the option to match the color themes of the graphs to one's research and making the interface more pleasant to use in dark environments.

Validation of results

To verify the validity of the application, results from other publications were replicated. In essence, the ZenBand solver can be divided into two types, even though they are based on the same set of equations—one for band diagrams (or iso-frequency contours) and one for field distributions. Another integral part of the program is geometry—square and hexagonal (oblique) symmetries require correctly defined lattice vectors in both real and reciprocal space.

Figure 4a compares ZenBand's band diagram to a diagram that was published in literature[40]. A structure made of dielectric columns with radii of $r = 0.2a$ and dielectric permittivity of $\varepsilon_r = 8.9$ was modeled in order to explain the appearance of band gaps in equivalent lattices. When the band gap widths for E mode were compared between ZenBand and the publication, only a 1.7% disagreement was found. This can be attributed to slightly different grid sizes of the unit cell or the number of spatial harmonics used in the calculations.

The validity of hexagonal symmetry in the application was verified by recreating a band diagram from a publication that investigated the possible uses of 2D PhCs for waveguides[37]. Once again, a band gap analysis for a dielectric waveguide with air holes was done ($r^* = 0.43a$, $\varepsilon_r = 11.6$, Fig. 4b). ZenBand results match the publication almost exactly in this case, there being no difference between the OBG width for E mode to the accuracy of the third digit after the decimal point and only a 1.6% disagreement for H mode.

Hexagonal symmetry was also inspected by repeating iso-frequency contour graphs that were taken from an article where an improvement in collimation effect for dielectric cylinders in a hexagonal lattice was demonstrated by breaking rotational symmetry[38]. The results were compared with symmetrical lattices ($r^* = 0.2a$, $\varepsilon_r = 9.8$)

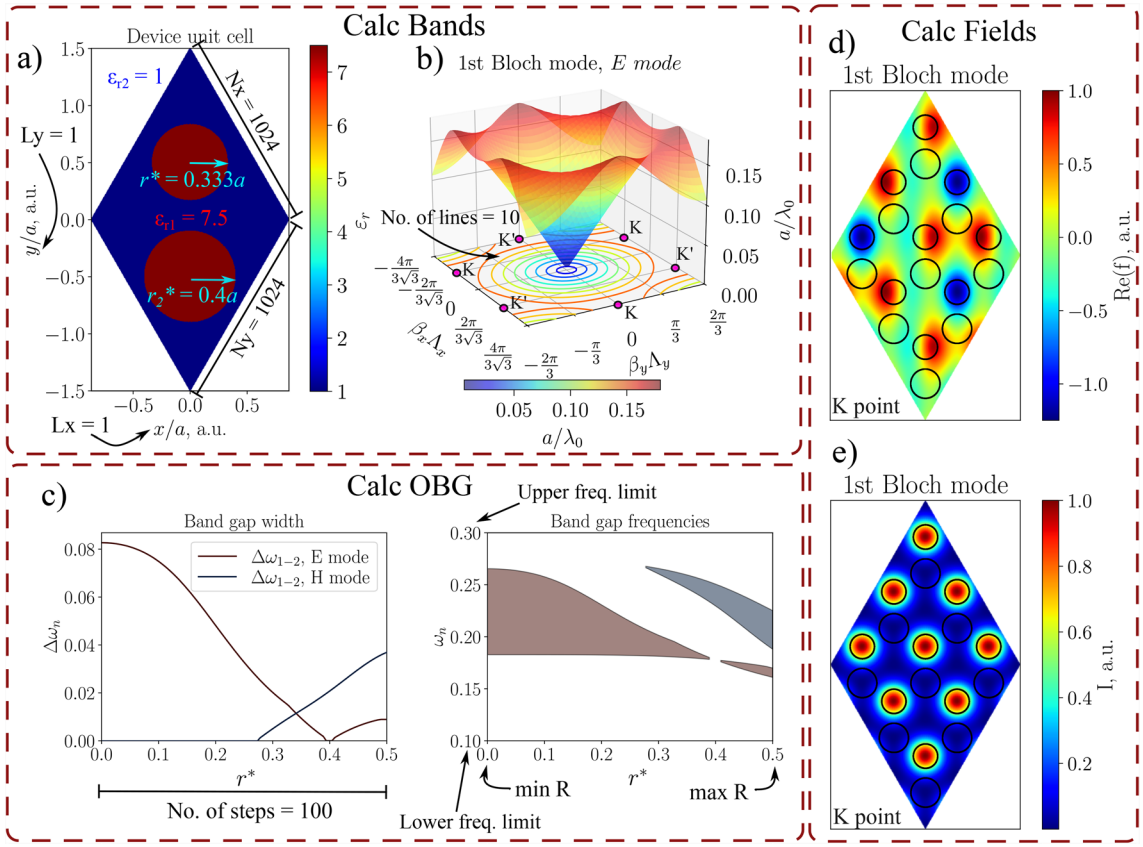

Fig. 3. Different functions of ZenBand. The purpose of 'Calc Bands' tab is to display the unit cell (**a**) and calculate photonic band diagrams, such as 3D iso-frequency contours (**b**). 'Calc OBG' tab is used for omnidirectional band gap analysis—the output of 'Param sweep for R' command is displayed in (**c**) where the value of r^* was changed from 0 to $0.5a$ and r_2^* was kept constant with a value of $0.4a$. 'Calc Fields' tab lets the user output field distributions of the device. Output of 'Calc Fz' command is shown in (**d**) for the real part and (**e**) for the intensity of the field in K point in the reciprocal lattice.

whose iso-frequency contours are depicted in Fig. 4c. Green stars represent contours in the publication that have the same frequency values as the ZenBand graph. It is clearly visible that the stars and contours overlap, providing nearly identical results.

To validate the field distribution capabilities of ZenBand, we investigated a honeycomb lattice beam splitter structure previously modeled using dielectric rods [39]. The reference work specifies a normalized radius of $r^*_{\sqrt{3}} = 0.1433a$. It is important to note that this value implies a normalization factor of $\sqrt{3}$; for clarity and consistency, we define the standard radius as $r^* = 0.248a$ ($\approx 0.1433a \times \sqrt{3}$) and set $\varepsilon_r = 12$ within ZenBand.

Our calculations reveal a Dirac point consisting of three bands, spanning Bloch modes 5–7. Figure 4d compares the field distributions computed by ZenBand with the FEM-based results from COMSOL Multiphysics [39] (shown in the bottom panels). While the symmetries and dipole orientations align perfectly, some amplitude and phase discrepancies exist. These deviations are characteristic of the Plane Wave Expansion Method (PWEM) used in ZenBand.

Such a discrepancy is due to the nature of the inverse Fourier transform used in the model (see 21-line of Algorithm 1). The algorithm centers the Fourier field's coefficients at the zeroes of a finite distribution of spatial harmonics, after which the inverse Fourier Transform is taken—this is technically called "zero-padding"[41]. Also, Gibbs[42,43] phenomena occur near steep permittivity gradients, resulting in minor field distortions that do not affect the fundamental modal symmetries .

Figure 4e displays the recreated band diagram with the Dirac point at the Γ point, while Fig. 4f illustrates the iso-frequency contours. Regarding frequency normalization, the reference work scales eigenvalues to frequency, whereas we utilize the normalized frequency ratio a/λ_0. We consider the latter more practical for geometric scaling in device design. Under this convention, our calculated Dirac point lies at $a/\lambda_0 = 0.4448$, corresponding well with the reference value of $a_{\sqrt{3}}/\lambda_0 = 0.4442$ (a deviation of 0.13%).

Comparison between environments and convergence

The Fourier expansion-based algorithms and their optimized implementations ensure the fastest possible computation of eigenvalues and eigenvectors of a periodic system. This particularly involves the PWEM and Rigorous Coupled Wave Analysis (RCWA) algorithms for device scattering simulations[44]. The main result of the

Fourier expansion of the fields is that the resolution of eigen-matrices depends only on the spatial harmonics used in the mentioned algorithms and thus can be implemented for low to medium contrasts of dielectric gratings, photonic crystals, etc. The outcome is increased computational speed and reduced memory requirements to perform tasks. Our goal is to compare and evaluate the performance of Python, MATLAB, and compiled C++ in terms of computing eigenvalue problems. The speed comparison (in seconds) is illustrated in Fig. 5a,b.

For implementation of the same algorithm in numpy, ordinary eigenvalue solver *np.linalg.eigs* outperforms Matlab for TE polarization (the H mode), because the permeability tensor is unity. In this case, Matlab performs 1.25–2 times slower starting from 225 harmonics (P=Q=15), where ratio= $t(\mathrm{Python})/\mathrm{t(Matlab)}$ is used. However, the main issue here is the generalized eigenvalue solver, which is implemented using SciPy. It is built with the numpy library, rather than being compiled with C language. Unfortunately, it comes with a cost of computational speed, and thus underperforms compared to Matlab by a factor of 15–20 (see Fig. 5b) for E mode (TM polarization). The time capture is performed for a single β iteration. The boost of performance can be obtained by solving such a generalized eigen-value problem either with a GPU (CuPy), assuming more than $\mathrm{NH} = 500$ spatial harmonics are used, or if the spatial harmonic number is lower, then the Scipy's linear algerba function 'eigh' for generalized hermitian matrices can be utilized.

The performance of the algorithm was examined by studying the convergence. More precisely, the change in error was checked for different numbers of spatial harmonics. In this case, error is defined as:

$$error = \sqrt{\left(\frac{\omega_n^{25\cdot 25} - \omega_n^{P\cdot Q}}{\omega_n^{25\cdot 25}}\right)^2}, \tag{3}$$

where ω_n^{25x25} is the normalized frequency value for *n*th band with $P = Q = 25$ spatial harmonics (which could be considered highly accurate for circular patterns in a lattice) and $\omega_n^{P\cdot Q}$ is the normalized frequency value for the same band with *P* and *Q* numbers of spatial harmonics in *x* and *y* directions.

The convergence test was performed on a hexagonal lattice with dielectric permittivity of $\varepsilon_r = 12$ and $r^* = 0.4a$ radius air holes ($\varepsilon_r = 1$). The frequencies were analyzed at *K* point of the reciprocal lattice. The results are displayed in Fig. 5c,d. Graph (c) shows how the error changes for different numbers of spatial harmonics. Graph (d) shows how many harmonics are needed for different order bands (1st to 15th) to reach convergence, which is an error value less than 0.01%.

It can be observed in Fig. 5c that the error reduces a bit faster for H mode when the number of harmonics is low (< 100), but for a higher number of *PQ*, E mode starts to converge faster. This can be explained as follows. H mode does not require a generalized eigenvalue equation, and it converges slightly faster than E mode, but only up to a certain point. Once *PQ* reaches a large enough value, $[\![\varepsilon_r]\!]^{-1}$ becomes increasingly more complex and induces more numerical errors. On the other hand, since the material is non-magnetic, $[\![\mu_r]\!]^{-1}$ stays unity. The same idea can be used to explain Fig. 5d—the change of spatial harmonics for convergence is straightforward for E mode, as it increases with the number of the band. However, H mode has a less predictable dependence. Such dependence arises when the sum of two Fourier expansions that are discontinuous at the same point is multiplied together. Such a case can be found for the normal component of the electric field at the interface of a dielectric $\epsilon_r \cdot \vec{E}$. It is possible to handle such a problem, but it requires a normal vector matrix for each

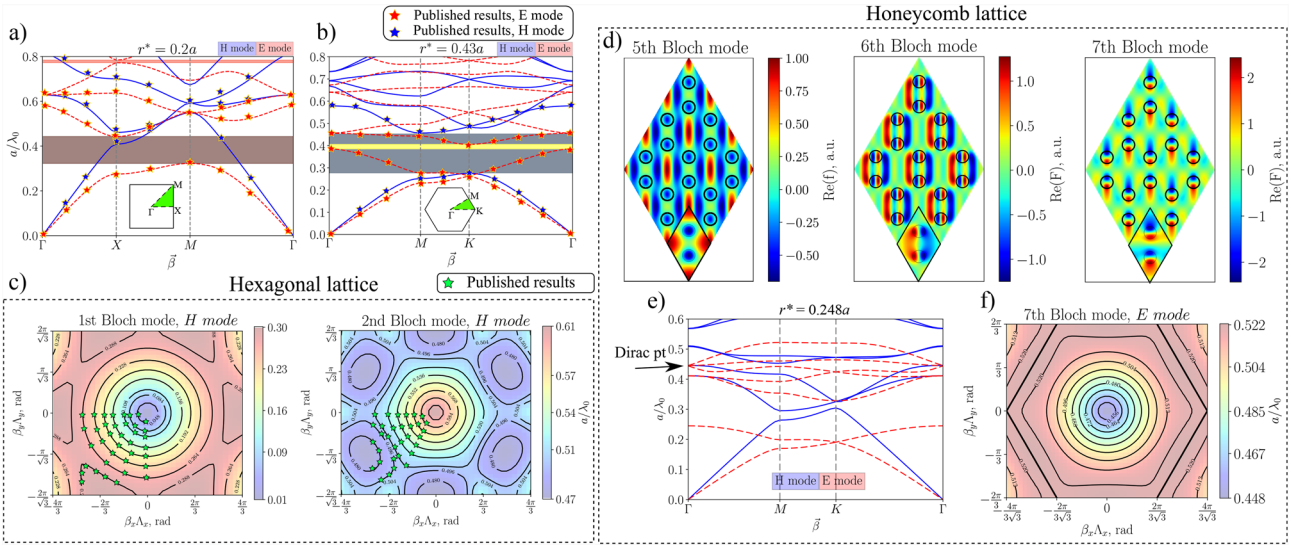

Fig. 4. Cross-validation of the ZENBAND. The (**a,b**) parts compare band diagrams for square[36] and hexagonal[37] lattices, respectively. (**c**) Recreates iso-frequency contours from[38]. Field distributions for the honeycomb lattice, calculated with ZenBand, are shown in (**d**) and compared with[39]. (**e,f**) The band diagram with the Dirac point and iso-frequency contours from the publication.

individual unit cell, which is a non-trivial task and would significantly impact the calculation speed. The method for such handling is abbreviated as Fast Fourier Factorization[45].

However, the PWEM must also be tested for extreme values of permittivity. The convergence test is presented in Fig. 6 for ZenBand's eigenvalue solver under those extreme conditions. It was tested using dielectric rods in air and a dielectric slab with air holes in a hexagonal lattice. Two cases were analyzed: when relative dielectric permittivity is $\varepsilon_r = 20$ [graphs (a) and (b)] and $\varepsilon_r = 100$ [graphs (d) and (e)]. Figure 6c depicts the $\varepsilon_r = 20$ case for 1st and 11th Bloch modes, including both the TE and TM polarizations. Up to NH $=33 \cdot 33$ spatial harmonics were used to complete the simulation. The strong flattening of the photonic bands for E mode (cylinder case) and H mode (air hole case) specifies that their fields are strongly localized within the PhC structure. The strong localization of the E_z field is within rods for high ϵ_r, whereas for air hole distribution the strongest localization of the H_z field is within the veins[40].

Conclusions

ZenBand is an open-source 2D PWEM solver with a simple graphical user interface that can be installed as a Python package. The program can be used to calculate photonic band diagrams and iso-frequency contours, analyze omnidirectional band gaps, and visualize Bloch fields. The application is easy to use, even for inexperienced or nonspecialized researchers, as it includes some of the most commonly found and widely applicable geometries. More experienced users have the opportunity to import their own devices and geometries if they desire to analyze more unusual or novel lattices.

Both the PWEM solver and the geometry of the lattices in ZenBand were validated by recreating graphs from various publications. The results are almost entirely consistent with other research, and the minor discrepancies can be attributed to numerical errors. The performance of the solver was also compared between Python and MATLAB environments. When the generalized eigenvalue solver is not used, Python performs exceptionally well and can be compared to MATLAB. Once the generalized eigenvalue solver is used, performance drops significantly; however, it remains sufficiently fast and outperforms other open-source solvers.

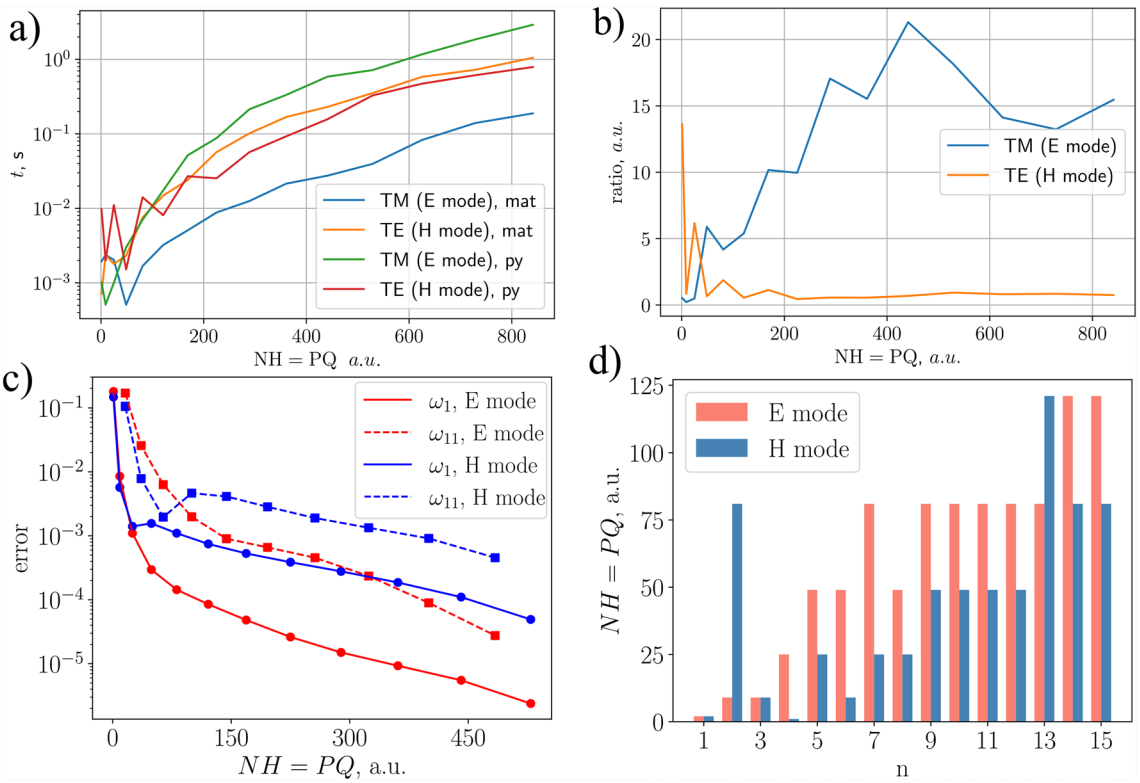

Fig. 5. Direct time comparison (**a**) (in seconds) vs spatial harmonics, given P=Q. The ratio between Python numpy and MATLAB environments (**b**). The convergence test for a hexagonal lattice is depicted in graphs (**c,d**). Graph on the left (**c**) shows the error dependence on the number of spatial harmonics for the 1st and 11th bands for both E and H modes. The graph on the right (**d**) indicates the number of spatial harmonics needed to reach convergence for 1st–15th bands (denoted as *n* on the *x* axis). The convergence condition was set that the error must be lower than 0.01 %.

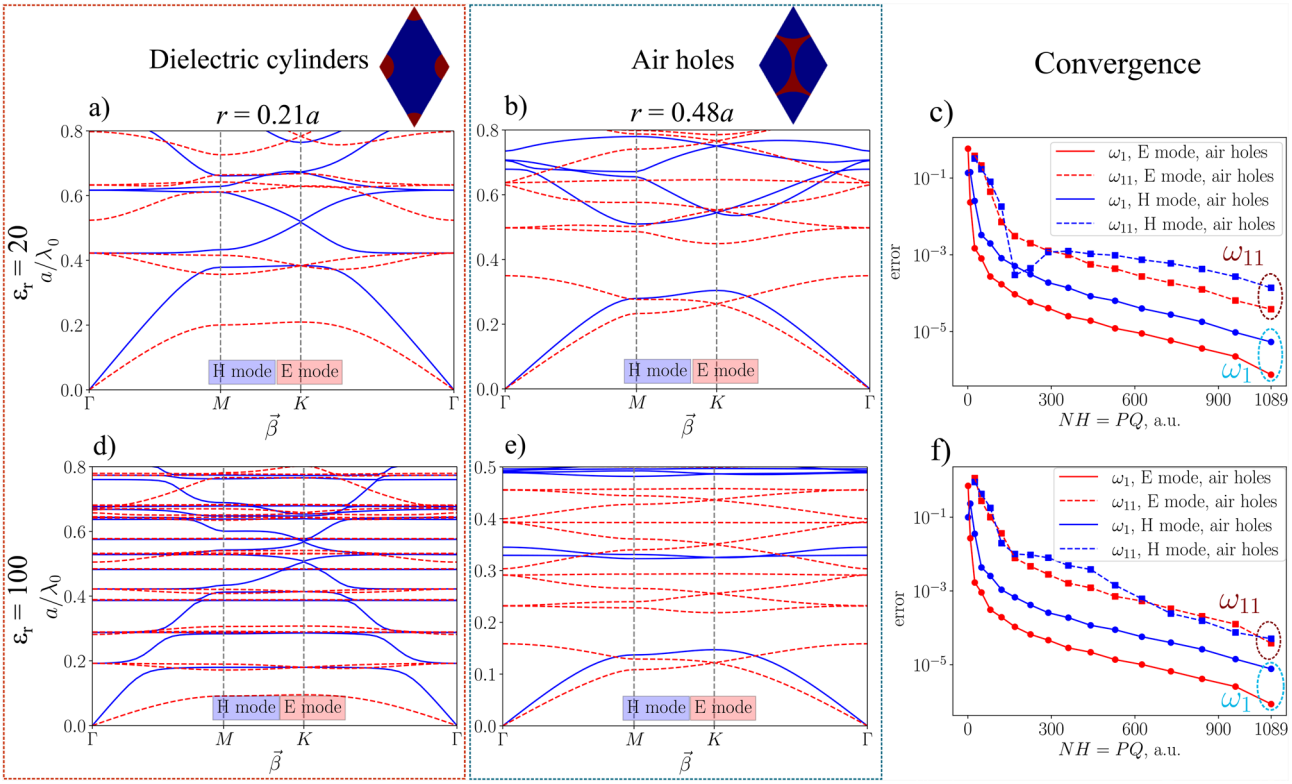

Fig. 6. Convergence test under extreme conditions. (**a,b**) Depict dielectric cylinders and air holes as hexagonal arrays, with $\varepsilon_r = 20$. Graph (**c**) shows the convergence error of eigenvalue, based on (3) for 1st and 11th Bloch modes, including both polarizations. Graphs (**d,e**) are equivalent to (**a,b**) but the material's relative permittivity $\varepsilon_r = 100$. Part (**f**) gives the convergence for the air hole distribution with the latter dielectric constant.

Data availability

The datasets used and/or analyzed during the current study are available from the corresponding author on reasonable request.

Received: 26 September 2025; Accepted: 20 January 2026
Published online: 04 February 2026

References

1. Tang, Y. et al. Optical neural engine for solving scientific partial differential equations. *Nat. Commun.* **16**, 4603 (2025).
2. Vial, B. & Hao, Y. Open-source computational photonics with auto differentiable topology optimization. *Mathematics* **10**, 3912 (2022).
3. Jiang, X. et al. Physics-informed neural network for nonlinear dynamics in fiber optics. *Laser Photon. Rev.* **16**, 2100483 (2022).
4. Su, Y. & Zhang, Q. Glare: A free and open-source software for generation and assessment of digital speckle pattern. *Opt. Lasers Eng.* **148**, 106766 (2022).
5. Neese, F. Software update: The orca program system—version 5.0. *Wiley Interdiscip. Rev. Comput. Mol. Sci.* **12**, e1606 (2022).
6. Erdem, S. et al. Rayx—An optics simulation software for synchrotron applications. *Rev. Sci. Instrum.* **96** (2025).
7. Woodhams, L. G. et al. Virtual Blebbistatin: A robust and rapid software approach to motion artifact removal in optical mapping of cardiomyocytes. *Proc. Natl. Acad. Sci.* **120**, e2212949120 (2023).
8. Lamb, J. R., Ward, E. N. & Kaminski, C. F. Open-source software package for on-the-fly deskewing and live viewing of volumetric lightsheet microscopy data. *Biomed. Opt. Exp.* **14**, 834–845 (2023).
9. Gröhl, J. et al. Simpa: An open-source toolkit for simulation and image processing for photonics and acoustics. *J. Biomed. Opt.* **27**, 083010–083010 (2022).
10. Oskooi, A. F. et al. MEEP: A flexible free-software package for electromagnetic simulations by the FDTD method. *Comput. Phys. Commun.* **181**, 687–702. https://doi.org/10.1016/j.cpc.2009.11.008 (2010).
11. Kim, C. & Lee, B. Torcwa: GPU-accelerated Fourier modal method and gradient-based optimization for metasurface design. *Comput. Phys. Commun.* **282**, 108552. https://doi.org/10.1016/j.cpc.2022.108552 (2023).
12. Yoon, G. & Rho, J. Maxim: Metasurfaces-oriented electromagnetic wave simulation software with intuitive graphical user interfaces. *Comput. Phys. Commun.* **264**, 107846. https://doi.org/10.1016/j.cpc.2021.107846 (2021).
13. Sanchez-Brea, L. M. et al. Diffraction: An open-source library for diffraction and interference calculations. In *Optics and Photonics for Advanced Dimensional Metrology III* (de Groot, P. J., Guzman, F. & Picart, P. eds.) . Vol. 12997. 129971B. https://doi.org/10.1117/12.3021879. (International Society for Optics and Photonics, SPIE, 2024).
14. Dharmavarapu, R., Ng, S. H., Eftekhari, F., Juodkazis, S. & Bhattacharya, S. Metaoptics: Opensource software for designing metasurface optical element GDSII layouts. *Opt. Exp.* **28**, 3505–3516. https://doi.org/10.1364/OE.384057 (2020).
15. Dharmavarapu, R., Bhattacharya, S. & Juodkazis, S. GDOESII: Software for design of diffractive optical elements and phase mask conversion to GDSII lithography files. *SoftwareX* **9**, 126–131. https://doi.org/10.1016/j.softx.2019.01.012 (2019).

16. Veettikazhy, M. et al. Bpm-Matlab: An open-source optical propagation simulation tool in Matlab. *Opt. Exp.* **29**, 11819–11832. https://doi.org/10.1364/OE.420493 (2021).
17. Magalhães, T. E. & Rebordão, J. M. Pywolf: A pyopencl implementation for simulating the propagation of partially coherent light. *Comput. Phys. Commun.* **276**, 108336. https://doi.org/10.1016/j.cpc.2022.108336 (2022).
18. Loetgering, L. et al. Ptylab.m/py/jl: A cross-platform, open-source inverse modeling toolbox for conventional and Fourier ptychography. *Opt. Exp.* **31**, 13763–13797. https://doi.org/10.1364/OE.485370 (2023).
19. Hugonin, J. P. & Lalanne, P. Reticolo software for grating analysis. *arXiv preprint* arXiv:2101.00901 (2021).
20. Zuo, S., Doñoro, D. G., Zhang, Y., Bai, Y. & Zhao, X. Simulation of challenging electromagnetic problems using a massively parallel finite element method solver. *IEEE Access* **7**, 20346–20362 (2019).
21. Yavich, N. & Zhdanov, M. S. Finite-element EM modelling on hexahedral grids with an FD solver as a pre-conditioner. *Geophys. J. Int.* **223**, 840–850 (2020).
22. Liu, V. & Fan, S. S4: A free electromagnetic solver for layered periodic structures. *Comput. Phys. Commun.* **183**, 2233–2244 (2012).
23. Tang, G.-J. et al. Broadband and fabrication-tolerant 3-db couplers with topological valley edge modes. *Light Sci. Appl.* **13**, 166 (2024).
24. Huang, H., Yang, C., Li, H. & Zhang, Z. Unveiling the potential of photonic crystal surface emitting lasers: A concise review. In *Semiconductor Science and Technology* (2025).
25. Han, C. et al. Generating first-order optical vortex beams by photonic crystal slabs. *Opt. Exp.* **32**, 27591–27598 (2024).
26. Deng, R. et al. Broadband complete polarization control via inverse-designed photonic crystal slabs. *Adv. Opt. Mater.* **12**, 2303218 (2024).
27. Ho, K., Chan, C. T. & Soukoulis, C. M. Existence of a photonic gap in periodic dielectric structures. *Phys. Rev. Lett.* **65**, 3152 (1990).
28. Rumpf, R., Garcia, C., Berry, E. & Barton, J. Finite-difference frequency-domain algorithm for modeling electromagnetic scattering from general anisotropic objects. *Prog. Electromagnet. Res. B* **61**, 55–67. https://doi.org/10.2528/PIERB14071606 (2014).
29. Rumpf, R. C. Engineering the dispersion and anisotropy of periodic electromagnetic structures. In *Solid State Physics*. Vol. 66. 213–300 (Elsevier, 2015).
30. Zanotti, S. et al. Legume: A free implementation of the guided-mode expansion method for photonic crystal slabs. *Comput. Phys. Commun.* **304**, 109286 (2024).
31. Yang, Y., Jiang, H. & Hang, Z. H. Topological valley transport in two-dimensional honeycomb photonic crystals. *Sci. Rep.* **8**, 1588 (2018).
32. Shi, B. et al. Efficient coupling of topological photonic crystal waveguides based on transverse spin matching mechanism. *Nat. Commun.* **16**. https://doi.org/10.1038/s41467-025-59941-6 (2025).
33. Li, M. et al. Ultrabroadband valley transmission and corner states in valley photonic crystals with dendritic structure. *Commun. Phys.* **7**, 214 (2024).
34. Xu, Z.-H. et al. Quantitative terahertz communication evaluation of compact valley topological photonic crystal waveguides. *ACS Photon.* **12**, 1822–1828 (2025).
35. Xu, X. et al. Polymer valley photonic crystals with honeycomb structures for terahertz waveguides. *Opt. Commun.* **538**, 129446 (2023).
36. Moniem, T. A. All optical active high decoder using integrated 2D square lattice photonic crystals. *J. Mod. Opt.* **62**, 1643–1649 (2015).
37. Jamois, C. et al. Silicon-based two-dimensional photonic crystal waveguides. *Photon. Nanostruct.-Fundam. Appl.* **1**, 1–13 (2003).
38. Yuksel, Z. M. et al. Enhanced self-collimation effect by low rotational symmetry in hexagonal lattice photonic crystals. *Phys. Scr.* **99**, 065017 (2024).
39. Qiu, P. et al. Investigation of beam splitter in a zero-refractive-index photonic crystal at the frequency of Dirac-like point. *Sci. Rep.* **7**. https://doi.org/10.1038/s41598-017-10056-z (2017).
40. Meade, R. D. V., Johnson, S. G. & Winn, J. N. Photonic crystals: Molding the flow of light (2008).
41. Oppenheim, A. V. & Schafer, R. W. *Discrete-Time Signal Processing*. Prentice-Hall Signal Processing Series (Pearson, 2010).
42. Arik, K. & Akbari, M. Polynomial expansion method for full wave three-dimensional analysis of dielectric waveguides and periodic structures. *Opt. Exp.* **32**, 16115–16131. https://doi.org/10.1364/OE.519283 (2024).
43. Fay, T. H. & Kloppers, P. H. The Gibbs' phenomenon. *Int. J. Math. Educ. Sci. Technol.* **32**, 73–89. https://doi.org/10.1080/00207390117151 (2001).
44. Lou, B. & Fan, S. Rcwa4d: Electromagnetic solver for layered structures with incommensurate periodicities. *Comput. Phys. Commun.* **306**, 109356. https://doi.org/10.1016/j.cpc.2024.109356 (2025).
45. Götz, P., Schuster, T., Frenner, K., Rafler, S. & Osten, W. Normal vector method for the RCWA with automated vector field generation. *Opt. Exp.* **16**, 17295–17301. https://doi.org/10.1364/OE.16.017295 (2008).

Author contributions

I.L. and A.Z. developed ZenBand academic and commercial software, D.G. is the leader of the technical project. All authors contributed equally to this paper. I.L. and A.Z. have made equal contributions to the ZenBand software.

Funding

D.G. acknowledges funding from the Research Council of Lithuania (LMTLT), grant No. S-MIP-23-49. A.Z. and I.L. acknowledge the "Universities' Excellence Initiative" programme by the Ministry of Education, Science and Sports of the Republic of Lithuania under the agreement with the Research Council of Lithuania (project No. S-A-UEI-23–6).

Declarations

Competing interests

The authors declare no competing interests.

Additional information

Correspondence and requests for materials should be addressed to A.Z.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.