Blaze2D: A Custom Plane-Wave Eigensolver for Photonic Crystals

Rene-Marcel Lehner, Apl. Prof. Dr. Doris Reiter
TU Dortmund University, Condensed Matter Theory
rene-marcel.lehner@tu-dortmund.de, doris.reiter@tu-dortmund.de

July 6, 2026

Abstract

Blaze2D is a two-dimensional plane-wave expansion eigensolver written in Rust, developed to provide high-throughput Bloch-level data for envelope-approximation studies of photonic moiré crystals. As an alternative to MPB, it reproduces MPB's eigenvalues to within mixed-precision solver tolerance while reducing runtime by roughly a factor of two through mixed-precision arithmetic. The solver combines a locally optimal block preconditioned conjugate gradient (LOBPCG) eigensolver with analytic subpixel smoothing and direct TE/TM operator implementations, and it exposes Bloch coefficients and derived operator matrix elements directly to a Python pipeline. This paper summarizes the design, validation against MPB, and benchmarks of Blaze2D.

Keywords: photonic crystal, plane-wave expansion, LOBPCG, MPB, eigensolver, Rust, mixed precision

1 Introduction

Photonic crystals are periodic dielectric structures whose wavelength-scale modulation opens frequency gaps for electromagnetic waves. They have become a platform for slow light, topological photonics, and integrated optical devices. Their design relies almost entirely on numerical band-structure calculations, since closed-form solutions are restricted to a handful of idealized geometries. Among the available techniques, the plane-wave expansion (PWE) method occupies a privileged position for two-dimensional problems: it expresses the Bloch eigenproblem in a Fourier basis whose convergence is spectral, and it returns not only eigenfrequencies but the Bloch coefficients themselves, which are the raw data needed for any analysis that goes beyond the band diagram (group velocities, mode volumes, coupling matrix elements, Berry curvatures, and so on).

Blaze2D logo.

The two-dimensional eigenproblem. For a dielectric profile $\epsilon(\boldsymbol{r})$ that is periodic in the xy-plane and invariant along z, Maxwell's source-free equations decouple into two scalar polarizations. Writing the field as a Bloch mode of wavevector $\boldsymbol{k}$ and frequency ω, the master equations read

$$\text{TM } (E_z): \quad -\nabla^2 E_z(\boldsymbol{r}) = \left(\frac{\omega}{c}\right)^2 \epsilon(\boldsymbol{r})\, E_z(\boldsymbol{r}), \tag{1}$$

$$\text{TE } (H_z): \quad -\nabla \cdot \left[\epsilon^{-1}(\boldsymbol{r})\, \nabla H_z(\boldsymbol{r})\right] = \left(\frac{\omega}{c}\right)^2 H_z(\boldsymbol{r}). \tag{2}$$

Both are generalized Hermitian eigenproblems of the form $A\,\psi = \lambda\,B\,\psi$ with $\lambda = (\omega/c)^2$. Expanding the field and the (inverse) permittivity on the reciprocal lattice $\{\boldsymbol{G}\}$ turns the differential operators into matrices indexed by plane-wave labels $\boldsymbol{G}, \boldsymbol{G}'$; the resulting discrete operators are exactly the ones implemented by Blaze2D and listed explicitly in Sec. 4. The TM problem is diagonal in the kinetic block and carries the permittivity in B, while the TE problem absorbs the inverse permittivity into a dense A. This asymmetry has direct algorithmic consequences that reappear later in the discussion of convergence rates and preconditioner design.

What is already available. MPB [1] has been the de-facto reference implementation of PWE for photonic crystals for more than two decades. Its accuracy, its analytical subpixel smoothing of dielectric interfaces, and its block-iterative eigensolver are the standard against which any new solver must be measured. The original MPB paper states this explicitly: it uses "preconditioned block-iterative eigensolvers in a planewave basis" and compares preconditioned conjugate-gradient Rayleigh–quotient minimization with Davidson's method [1]. Among freely available tools, MPB remains the reference solver for this specific class of PWE band-diagram and eigenpair calculations. Blaze2D targets the same problem class, but is optimized for high-throughput two-dimensional parameter sweeps in which a moderate plane-wave cutoff is solved thousands of times for slightly varying geometry and the Bloch coefficients must be exposed through a Python pipeline.

Design intent of Blaze2D. Blaze2D was built to fill this niche. It is a two-dimensional PWE eigensolver written in Rust, distributed as open source on GitHub [2] and as a package on PyPI [3]. Three concerns shape its architecture, each addressed in a later section:

1. *Accuracy on par with MPB.* Analytic subpixel smoothing of the permittivity [4] and a faithful reimplementation of MPB's preconditioning strategy ensure that Blaze2D reproduces MPB's eigenvalues to within solver tolerance across a broad set of two-dimensional geometries (Sec. 8).

2. *Throughput at moderate resolution.* A mixed-precision arithmetic strategy stores all bulk arrays in single precision while promoting numerically sensitive reductions to double precision, yielding a near-clean 2× speedup with no measurable loss in eigenvalue accuracy (Sec. 7). This finding mirrors an independent 2023 study on dynamic-precision LOBPCG for electronic structure [5].

3. *Direct access to Bloch coefficients.* The TE and TM operators are implemented separately and exposed through a Python interface, so that derived quantities such as matrix elements, mode profiles, and Berry connections can be computed from the same operator objects used by the eigensolve (Sec. 10).

The remainder of this paper develops these three ideas. Section 2 states the high-level design choices. Section 3 treats the construction of the Fourier-space permittivity matrix. Section 4 writes the discretized TE and TM operators explicitly, building directly on Eqs. (1) and (2). Sections 5–6 discuss preconditioning and the LOBPCG inner loop. Section 7 describes the mixed-precision strategy. Section 8 compares Blaze2D against MPB on spectra, mode profiles, and resolution convergence, while section 9 reports runtime and memory benchmarks. Section 10 sketches the data-extraction layer that connects the solver to subsequent analyses.

2 Design Choices

Three foundational choices were made from the start, each driven by a concrete requirement of the envelope pipeline.

Method: plane-wave expansion. The PWE method discretizes the Bloch eigenproblem in reciprocal space by expanding both the dielectric function and the field in Fourier modes on the reciprocal lattice. For a given $\boldsymbol{k}$-point and resolution (number of retained plane waves N_{pw}), the problem becomes a dense $N_{\mathrm{pw}} \times N_{\mathrm{pw}}$ matrix eigenproblem. The reason for choosing PWE over real-space methods is direct: the envelope framework requires not just eigenvalues but Bloch functions and their Fourier coefficients, and the PWE basis delivers these natively. Additionally, PWE eigenvalues converge spectrally with resolution, providing high accuracy at moderate grid sizes [1].

Algorithm: LOBPCG. The locally optimal block preconditioned conjugate gradient (LOBPCG) method [6] solves for the lowest eigenpairs of a Hermitian operator without forming or factorizing the full matrix. It is iterative, works in blocks (solving for several eigenpairs simultaneously), and accepts a preconditioner. These properties make it the natural choice for the photonic PWE problem, where only the lowest bands are needed and a good preconditioner is available. MPB uses the same algorithmic family [1].

Language: Rust. The solver must run millions of operator applications across thousands of crystal configurations. Rust provides predictable memory layout, no garbage-collection pauses, and safe concurrency. These properties matter once the solver runs inside a tight parameter-sweep loop. The compiled performance is comparable to C/Fortran, while the type system eliminates entire classes of memory bugs that would otherwise surface only under heavy workloads.

3 Geometry and the Permittivity Matrix

The first module to build is the *geometry*: translating a crystal specification (lattice vectors, shapes, dielectric constants) into the Fourier-space permittivity matrix $\epsilon_{\boldsymbol{G},\boldsymbol{G}'}$ processed by the PWE operators.

Blaze2D uses the same analytical subpixel smoothing as MPB. Rather than sampling $\epsilon(\boldsymbol{x})$ on a pixel grid, each geometric primitive (cylinder, ellipse, block) is treated as an exact shape whose overlap with every grid cell is computed analytically. The smoothed ϵ^{-1} is then obtained via the anisotropic averaging procedure of Farjadpour *et al.* [4], which avoids the staircasing artifacts that naive pixelization introduces at dielectric interfaces.

Validating this module is straightforward: the permittivity matrix can be compared element-by-element against MPB for any crystal geometry. Figure 1 shows such a comparison for a square lattice of dielectric rods ($\epsilon = 8.9$, $r = 0.45a$) at resolutions $N = 8$ to 128. Only the coarsest grid ($N = 8$) exhibits appreciable error (mean $|\Delta\epsilon| \approx 3 \times 10^{-2}$, peak ≈ 0.68); for $N \geq 16$ the mean error drops below 8×10^{-3} and reaches $\approx 5 \times 10^{-4}$ at $N = 128$. The remaining residual peaks of order 0.1–0.2 are localized at the rod–air interface, where the two solvers discretize the material boundary slightly differently. Agreement to this level at every resolution confirms that the geometry pipeline is correct before any eigensolve is attempted.

With the geometry validated, the remaining task is a spectral analysis: construct the TE and TM operators on a Fourier basis and solve them efficiently.

4 Eigenvalue Operators and Their Application

With the permittivity matrix in hand, the next step is to discretize the TE and TM master equations of the photonic-crystal eigenproblem explicitly. In contrast to MPB, which works from the full curl–curl operator and projects onto the relevant polarization, Blaze2D implements the

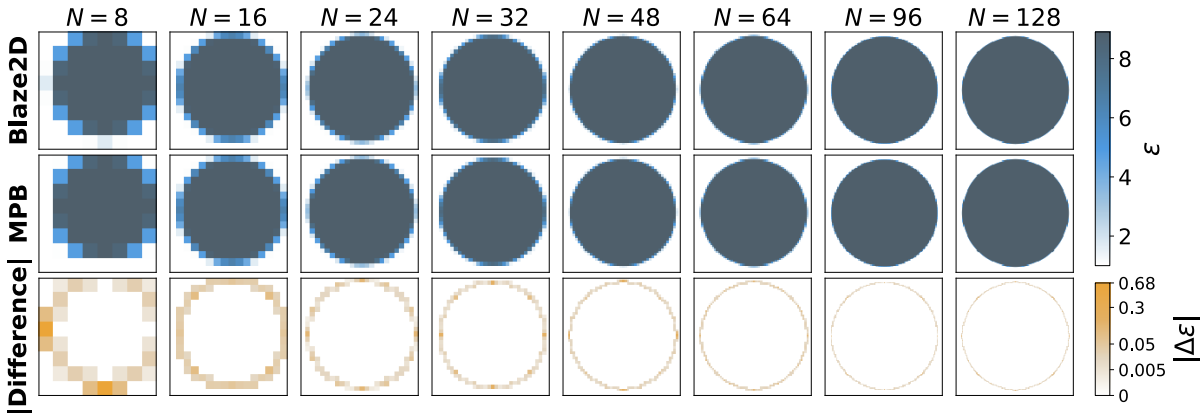

Figure 1: Smoothed permittivity maps from Blaze2D and MPB at resolutions $N = 8$ to 128, with the bottom row showing the absolute difference $|\Delta\epsilon|$ on a power-law color scale ($\gamma = 0.3$) to magnify small discrepancies.

TE and TM operators directly:

$$\text{TE:} \quad A_{\text{TE}} = -(\boldsymbol{k}+\boldsymbol{G}) \cdot [\epsilon^{-1}]_{\boldsymbol{G},\boldsymbol{G}'} (\boldsymbol{k}+\boldsymbol{G}'), \qquad B_{\text{TE}} = \mathbf{1}, \tag{3}$$

$$\text{TM:} \quad A_{\text{TM}} = |\boldsymbol{k}+\boldsymbol{G}|^2 \, \delta_{\boldsymbol{G},\boldsymbol{G}'}, \qquad B_{\text{TM}} = \epsilon_{\boldsymbol{G},\boldsymbol{G}'}. \tag{4}$$

The operator classes are implemented generically: the LOBPCG solver receives an abstract operator interface, making the TE and TM cases interchangeable at the API level.
Operator application proceeds in the standard PWE fashion. Multiplication by ϵ^{-1} in reciprocal space is a dense matrix–vector product, while the gradient terms $(\boldsymbol{k}+\boldsymbol{G})$ act as diagonal multiplications. In practice, these operations dominate the runtime and their memory-access pattern determines performance.
At high dielectric contrast, the Fourier coefficients of ϵ^{-1} decay slowly, and individual reciprocal-space components can develop large magnitudes that amplify numerical noise. Blaze2D applies the same clamping and regularization strategy as MPB to suppress these artifacts without distorting the physically relevant low-frequency content. This direct TE/TM implementation is also what makes the data extraction convenient: the same operator objects used in the eigensolve can later be reused to expose explicit matrix elements to the envelope pipeline.

5 Preconditioning

The convergence rate of LOBPCG depends critically on the preconditioner. Each polarization admits a different strategy.
For TE, the preconditioner is the one described in Ref. [1]: an approximation to A_{TE}^{-1} built from the diagonal kinetic term ($|\boldsymbol{k}+\boldsymbol{G}|^2$) and a scalar estimate of ϵ^{-1}. This is the same advanced preconditioner used by MPB and is the more complex of the two.
For TM, the structure is simpler. Because A_{TM} is already diagonal in the plane-wave basis, its inverse is trivially available and serves as an effective preconditioner. The result is that TM consistently achieves lower condition numbers and converges in fewer iterations than TE, a systematic advantage rooted in operator structure.
Tuning the preconditioner accounted for roughly a quarter of the total development time. The MPB developers describe the same challenge: finding a good preconditioner "is something of a *black art* with lots of trial and error"[1] [1]. Because Blaze2D's internal operator layout differs from MPB's, the preconditioner could not be ported directly; it had to be re-derived and re-tuned against condition-number measurements at each stage.

[1]From the MPB developer documentation, Developer Information: The Mathematics of MPB.

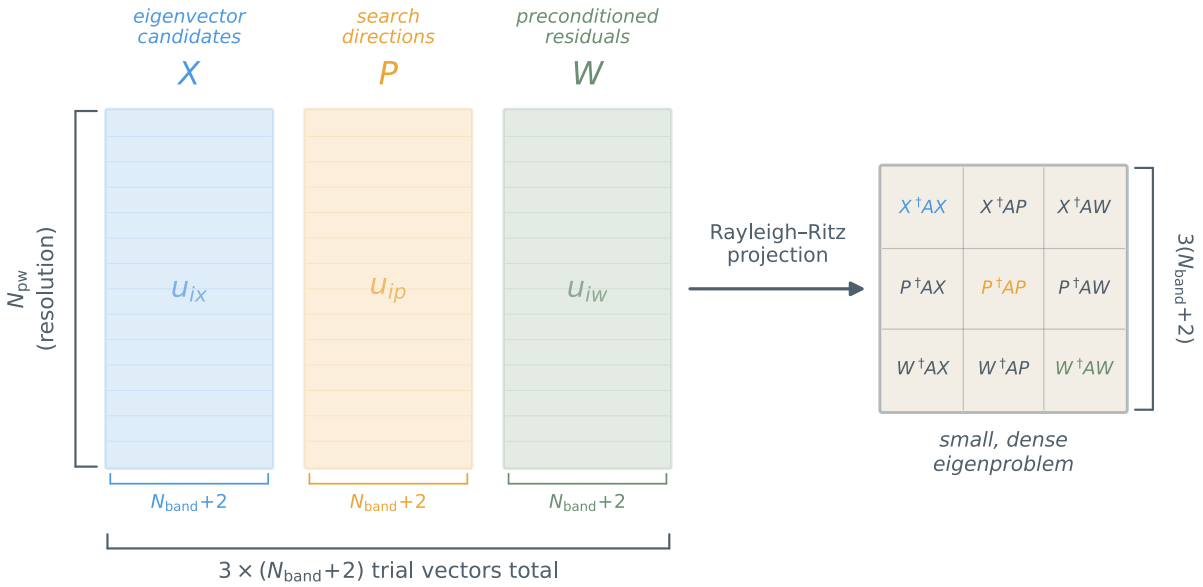

Figure 2: Structure of the LOBPCG search subspace. The trial basis $[X, P, W]$ defines a small, dense projected eigenproblem whose exact solution yields the updated eigenvector approximations. Rather than operating in the full N_{pw}-dimensional Hilbert space, each iteration works in a subspace of dimension $\dim(X) + \dim(P) + \dim(W)$.

6 The LOBPCG Solver

At the heart of Blaze2D lies the LOBPCG algorithm [6]. At each step, the solver maintains three subspaces: the current eigenvector candidates X (N_{band} vectors), the search directions P from the previous step, and the preconditioned residuals W. These are assembled into a trial basis $[X, P, W]$, and a small dense eigenproblem (the Rayleigh–Ritz projection) of dimension $\dim(X) + \dim(P) + \dim(W)$ is solved exactly. The new X is read off from the lowest eigenpairs of this projected problem. The structure of this search subspace is sketched in Fig. 2.
Several implementation details proved decisive for performance and stability:

Subspace dimensions. Expanding the candidate space beyond the number of requested bands accelerates convergence significantly: adding one or two extra trial vectors to X provides the optimizer with additional variational freedom at modest cost. Conversely, shrinking or omitting either P or W degrades convergence severely. The best results were achieved when all three subspaces had roughly comparable dimensions.

Orthogonalization. Near-degenerate eigenvectors cause classical Gram–Schmidt orthogonalization to lose numerical rank, producing a singular Rayleigh–Ritz problem. Blaze2D uses the SVQB procedure [7], which orthogonalizes via a singular-value decomposition of the Gram matrix and is numerically stable even for tightly clustered eigenvalues.

Convergence criterion. MPB monitors the residual norm $\|Ax - \lambda Bx\|$ and declares convergence when it falls below a threshold (default 10^{-7}). Blaze2D instead monitors the relative change in eigenvalues between iterations. In practice, eigenvalues stabilize well before the residual reaches a tight tolerance, so eigenvalue-based convergence requires substantially fewer iterations per solve. While residual convergence carries a proven convergence-rate guarantee [6], eigenvalue convergence proved completely stable across all tested configurations, including difficult cases with accidental degeneracies at high-symmetry $\boldsymbol{k}$-points.

Warm-starting. The single largest lever for performance is also the simplest: initializing each $\boldsymbol{k}$-point solve with the converged eigenvectors of the previous $\boldsymbol{k}$-point. This adds no algorithmic complexity, yet it typically reduces the iteration count by an order of magnitude compared to a random initialization. MPB uses the same strategy.

Deflation strategy. Once a band converges, the solver must avoid rediscovering it. Because distinct Bloch modes are mathematically orthogonal, we can explicitly project all new search directions away from the already-converged states, a technique known as deflation. While solvers can *hard-lock* these converged bands by removing them from the active problem entirely, Blaze2D relies on *soft-locking*. Converged bands remain part of the small Rayleigh–Ritz projection, but the solver stops calculating new, computationally expensive search directions for them. This halts their active optimization, yet allows the converged states to mix with the active subspace to maintain strict orthogonality as the remaining bands settle. In practice, soft-locking proved far more numerically stable than hard-locking, especially near accidental degeneracies where bands must be handled carefully as a collective cluster rather than frozen one by one.

7 Mixed Precision

Modern eigensolvers are not compute-bound but memory-bandwidth-bound. Profiling both MPB and Blaze2D shows that actual floating-point arithmetic accounts for as little as 10% of CPU time; the remainder is spent moving data between memory and cache. This observation motivates a mixed-precision strategy: store all large arrays (eigenvectors, operator matrices, intermediate products) in single precision (`f32`) to halve the memory traffic, but promote to double precision (`f64`) on the fly for numerically sensitive operations (inner products, Rayleigh–Ritz projections, and orthogonalization steps) before rounding the result back to `f32` for storage.
The natural objection is: if the result is rounded back to `f32`, what is gained by computing in `f64`? The answer is that certain intermediate accumulations require `f64` for numerical stability (a dot product of 10^4 terms, for instance, can lose several significant digits in `f32`), while the *result* of that dot product (a single scalar or a small matrix) is representable in `f32` without meaningful loss. The precision is needed transiently, not persistently.
This mixed-precision approach yields a nearly clean $2\times$ speedup with no significant loss in eigenvalue accuracy. The same conclusion was reached independently in a 2023 study of mixed-precision LOBPCG for electronic-structure calculations [5], a paper discovered only after the approach had already been implemented in Blaze2D. The strategy itself is standard in modern data science and high-performance computing.

8 Validation

Blaze2D was validated against MPB across a representative set of two-dimensional photonic crystals by comparing full band diagrams, $|\boldsymbol{u}|^2$ mode profiles, and convergence behavior. The validation process was the most time-consuming part of the project. MPB uses several nonstandard coordinate conventions, most notably a half-pixel-shifted grid center, which produced false discrepancies that had to be identified and accounted for one by one.
The validation narrative follows the same logic as the solver itself. First, the global spectral output must agree. Second, the eigenvectors must represent the same physical modes. Third, both solvers must converge in the same way as the discretization is refined. The following subsections address these three checks in turn.

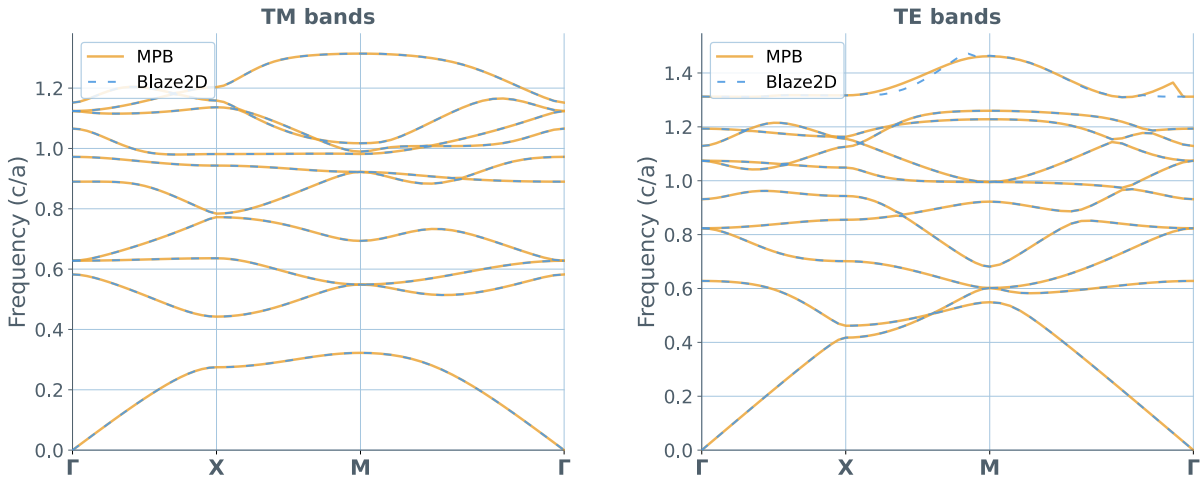

Figure 3: Band-diagram comparison for a square lattice ($\epsilon = 8.9$ rods, $r = 0.2a$), showing the 10 lowest bands. *Left*: TM polarization; *right*: TE polarization. Both solvers show near-perfect agreement, with small deviations visible only in the topmost TE band due to differing band-tracking behavior across avoided crossings.

8.1 Band-Diagram Agreement

The primary metric is the relative eigenvalue error across all $\boldsymbol{k}$-points and bands. Bloch-function phases were not compared directly, because MPB's phase conventions differ from Blaze2D's in ways that scramble the complex phases without affecting physically observable quantities. However, the squared modulus $|u_{n\boldsymbol{k}}(\boldsymbol{r})|^2$ was compared and found to agree, and improvements in mode-profile agreement tracked improvements in eigenvalue accuracy, confirming that the band diagram is a sufficient validation metric.

One systematic difference remains: when tracking bands across avoided crossings, MPB tends to follow a given band adiabatically to higher eigenvalues, while Blaze2D reports the true N lowest eigenvalues at each $\boldsymbol{k}$-point and does not track crossings. For isolated bands away from topmost-band crossings, both solvers agree to high precision. This behavior is visible in Fig. 3.

To quantify the agreement beyond the visual overlap of Fig. 3, Fig. 4 shows the RMS relative eigenvalue difference between the two solvers for the lowest eight bands, evaluated over 61 $\boldsymbol{k}$-points along $\Gamma \to X \to M \to \Gamma$, as a function of resolution. At $N = 128$, relative differences reach $\approx 2 \times 10^{-5}$ on average, with per-band maxima below 10^{-4}. This is consistent with Blaze2D's internal eigenvalue convergence tolerance of 10^{-5} in mixed-precision mode; the residual plateau beyond $N \approx 48$ reflects that tolerance floor rather than a systematic discrepancy between the two solvers.

Taken together, Fig. 3 and Fig. 4 establish a precise agreement: at the resolutions relevant for the registry sweep, Blaze2D reproduces MPB's eigenvalues to within the mixed-precision solver tolerance.

8.2 Mode Profiles

Eigenvalue agreement alone is not sufficient: the envelope pipeline ultimately processes eigenvectors and matrix elements built from them. For that reason, representative real-space mode profiles were compared as well. Figure 5 shows the squared Bloch amplitude $|\boldsymbol{u}|^2$ for several bands and $\boldsymbol{k}$-points.

The agreement is strong wherever the eigenvalue is non-degenerate: nodal structure, localization, and symmetry match visually across the unit cell. The only clearly visible mismatches occur at or near degeneracies, where any orthonormal basis of the degenerate subspace is equally valid. In those cases MPB and Blaze2D may return rotated or swapped representatives of the same

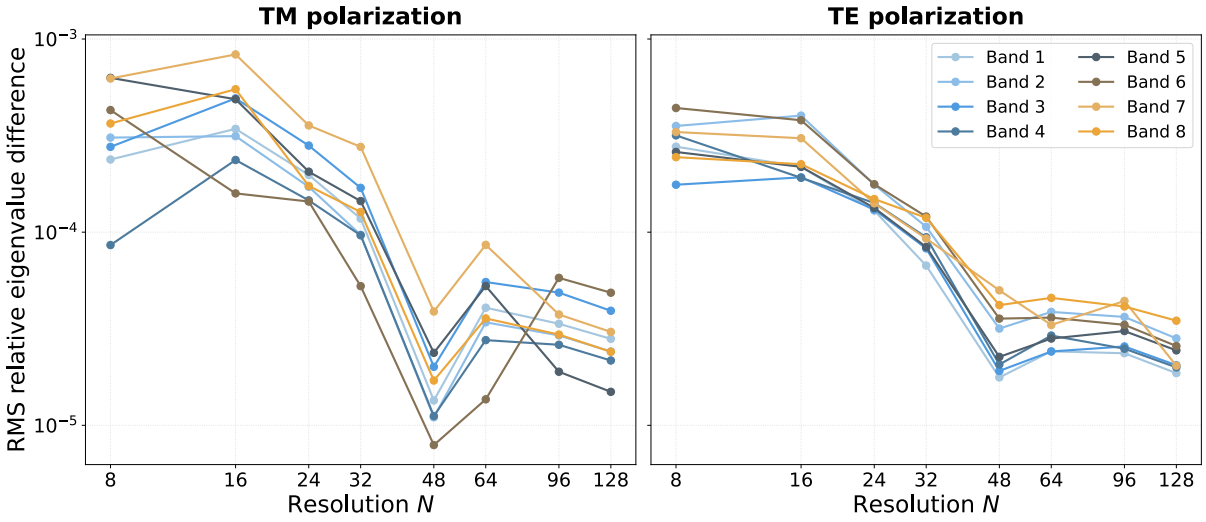

Figure 4: RMS relative eigenvalue difference between Blaze2D and MPB for the eight lowest bands of a square lattice ($\epsilon = 8.9$ rods, $r = 0.2a$). TM (left) and TE (right) polarizations.

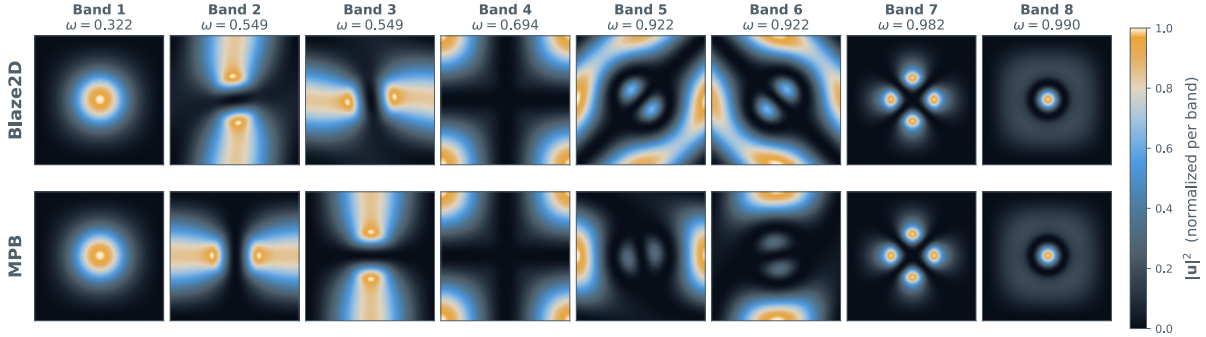

Figure 5: Comparison of $|\boldsymbol{u}|^2$ mode profiles from Blaze2D and MPB for a representative set of bands, $\boldsymbol{k}$-points and a resolution of 64 px/a. The spatial intensity distributions agree well for non-degenerate frequencies; at degenerate frequencies, the modes may be superposed or swapped, leading to expected visual discrepancies.

physical subspace, which changes the plotted pattern without indicating a physical disagreement. This is precisely why the comparison is carried out on $|\boldsymbol{u}|^2$ rather than on the complex Bloch phase itself.

Despite the strong agreement, small yet visual mismatches for bands 2 and 3 can be seen in Fig. 5. The spatial modes do not appear perfectly symmetric; these are deviations from MPB that have not been investigated or quantified here. Because the envelope pipeline is built from Bloch-function data, these deviations should be kept in mind when judging how reliable downstream envelope results are.

8.3 Convergence with Resolution

Agreement at a single resolution could still be accidental. A stronger test is whether both solvers approach the continuum limit at the same rate as the plane-wave cutoff is increased. That comparison is shown in Fig. 6.

The two curves track each other closely in both polarizations, for both RMS and worst-case errors. This matters because it rules out a hidden cancellation in the direct MPB comparison: Blaze2D does not merely match one reference calculation, but exhibits the same resolution dependence as an established solver. For the purposes of the envelope pipeline, this is the relevant notion of

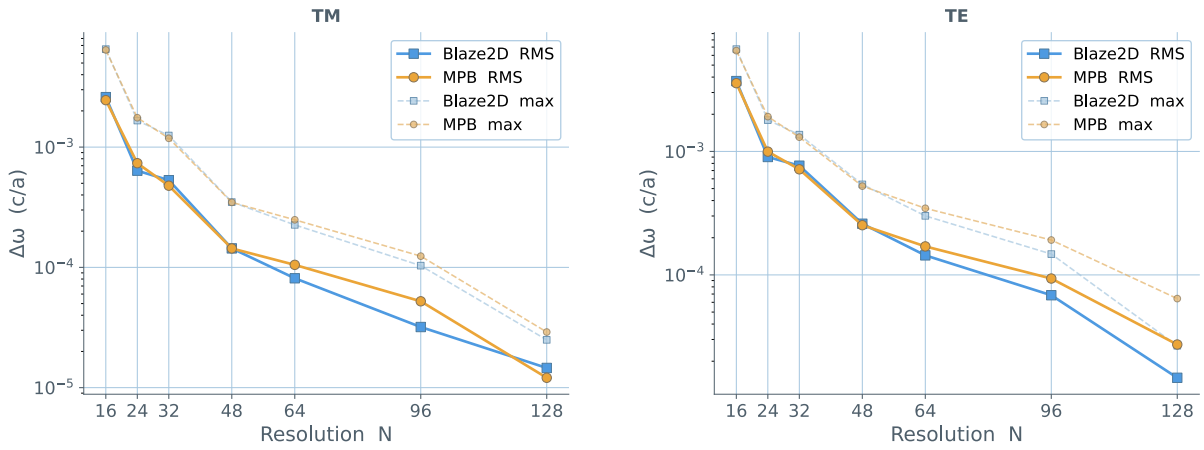

Figure 6: Eigenvalue convergence for a square lattice ($\epsilon = 8.9$ rods, $r = 0.2a$). Solid lines: RMS deviation $\Delta\omega_{\mathrm{RMS}}$ of the four lowest bands along $\Gamma \to X \to M \to \Gamma$; dashed lines: worst-case deviation $\max|\Delta\omega|$. All deviations are measured against a higher resolution $N = 192$ reference. Both solvers converge at comparable rates.

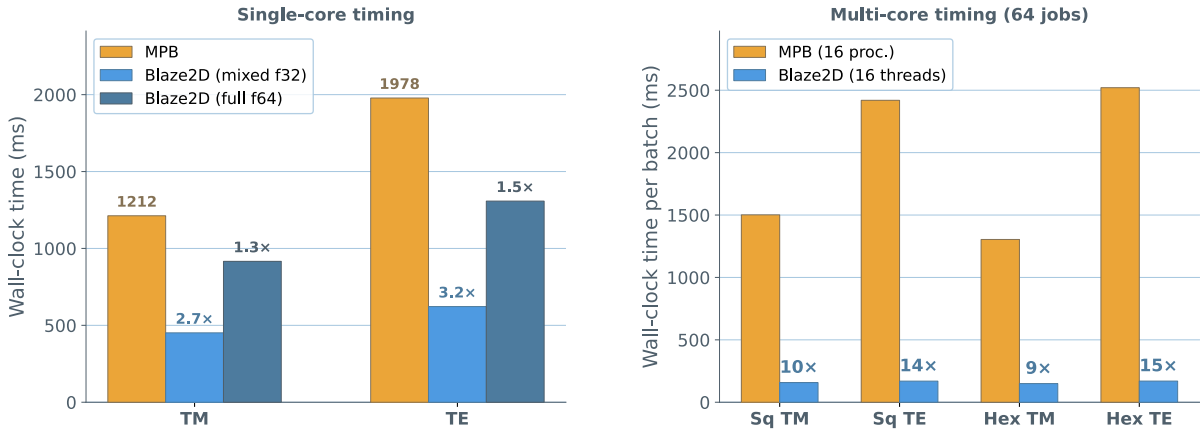

Figure 7: Single-core and multi-core wall-clock comparison of MPB and Blaze2D in mixed and full precision.

correctness, since all extracted matrix elements are ultimately built on these converged Bloch states.

9 Benchmarks

Accuracy alone would not justify a new solver. The point of Blaze2D is that it must also execute the dense registry sweep motivating it in reasonable time. The runtime comparison is shown in Fig. 7, while the iteration counts, per-iteration costs, and memory footprint are broken down in Fig. 8.

A few observations from these benchmarks deserve comment. The TM polarization converges in fewer iterations than TE for both solvers, consistent with the simpler preconditioner structure discussed above. Mixed precision does not increase the iteration count; the eigenvalue trajectory is indistinguishable from the full-precision run. The speedup seen in Fig. 7 therefore comes almost entirely from reduced memory traffic rather than from altered convergence behavior.

The decomposition in Fig. 8 makes this interpretation more concrete. Time per iteration falls under mixed precision, while memory consumption scales predictably with N_{pw}^2 for the stored matrices and $N_{\mathrm{pw}} \times N_{\mathrm{band}}$ for the eigenvector block, with `f32` storage halving both contributions

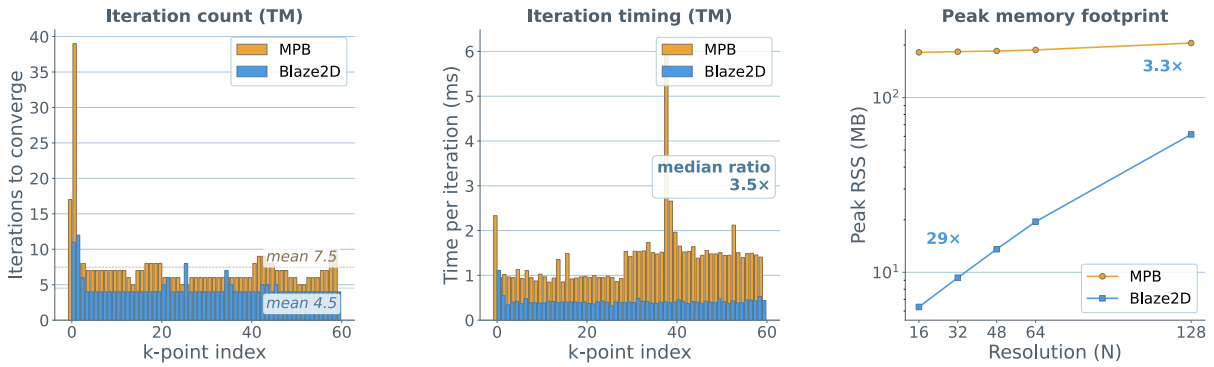

Figure 8: Iteration count per $\boldsymbol{k}$-point, time per iteration, and peak memory footprint for MPB vs. Blaze2D.

relative to `f64`. For the registry sweep, this reduction matters twice: it shortens each individual solve and increases the number of configurations that can be processed concurrently.

10 From Solver to Pipeline

With the eigensolver validated, the remaining task is to extract the matrix elements required by the envelope effective Hamiltonian and export them to the Python layer where the moiré-scale problem is assembled and solved.

The extraction module computes, for each crystal configuration $\boldsymbol{\delta}$ on the moiré grid, the local eigenvalues $\lambda_n(\boldsymbol{\delta}, \boldsymbol{k}_0)$, the Bloch functions $u_{n\boldsymbol{k}_0}(\boldsymbol{r}; \boldsymbol{\delta})$, and the derived operator matrix elements (Berry connections, effective-mass tensors, and the coupling terms entering the envelope effective Hamiltonian). These are written as structured arrays and exposed through Python bindings generated via Rust's foreign-function interface.

Basic sanity checks on the extracted data (Hermiticity of the effective Hamiltonian blocks, expected magnitudes of the Berry-connection components, correct symmetry under lattice operations) confirm that the plumbing between solver and pipeline introduces no artifacts. At this point the narrative closes back onto the envelope construction: the effective theory has been reduced to computable local data, and Blaze2D is the tool that provides it.

11 Conclusion

Blaze2D is a from-scratch two-dimensional plane-wave eigensolver that sets out to deliver MPB-grade accuracy with a workflow tuned for the high-throughput regime: many crystal configurations, moderate plane-wave cutoffs, and direct access to Bloch coefficients from a Python frontend. The construction proceeds along three areas.

Accuracy: analytic subpixel smoothing of the permittivity reproduces MPB's $\epsilon_{\boldsymbol{G},\boldsymbol{G}'}$ to within $\sim 10^{-4}$ in the mean even at the coarsest grids (Fig. 1); eigenvalues then agree to $\sim 2\times10^{-5}$ at the resolutions relevant for production runs (Fig. 4); and both solvers exhibit the same convergence rate as the plane-wave cutoff is refined (Fig. 6). The remaining discrepancies are localised at degenerate eigenvalues, where any orthonormal basis of the degenerate subspace is equally valid, or at dielectric interfaces, where the two solvers discretise the boundary slightly differently. Neither family is a numerical disagreement; both are explicable from solver conventions alone.

Throughput: the central performance lever is not algorithmic but architectural. Profiling reveals that LOBPCG on PWE operators is memory-bandwidth-bound, not compute-bound, so a mixed-precision storage strategy yields a near-clean 2× wall-clock speedup by storing bulk arrays in single precision and promoting only numerically sensitive reductions to double precision, without

altering the eigenvalue trajectory (Figs. 7 and 8). Rust's type-safe concurrency keeps multi-core scaling predictable inside parameter sweeps, where solves are independent and parallel.
Accessibility: implementing the TE and TM operators directly, rather than projecting them out of a general curl–curl operator, keeps the per-polarization data path short and makes operator-level matrix elements available to the Python layer through Rust's foreign-function interface. The same operator objects that drive the eigensolve are reused downstream, so derived quantities (group velocities, Berry connections, effective-mass tensors) are computed against exactly the discretization the eigenvalues were obtained on.
Several limitations should be acknowledged. Blaze2D is two-dimensional by design and currently exposes only the in-plane TE/TM decomposition; fully three-dimensional or out-of-plane band calculations require the more general curl–curl formulation and are left to upstream tools. Band tracking across avoided crossings is not attempted: the solver reports the N lowest eigenvalues at each $\boldsymbol{k}$-point without enforcing adiabatic continuity, which simplifies the inner loop but pushes the responsibility for band labelling onto the analysis stage. Small but visible asymmetries in degenerate mode profiles (Fig. 5) have not been quantified here; they are expected to be artefacts of the chosen basis within the degenerate subspace, but a rigorous gauge-fixing diagnostic remains future work.
Natural extensions follow from each of these limits. A three-dimensional curl–curl variant, an explicit band-tracking layer based on overlap matching across neighbouring $\boldsymbol{k}$-points, and a GPU back end that exploits the same mixed-precision idea on single-precision-favourable hardware are all plausible next steps. On the application side, exposing Bloch-level data through a stable Python ABI opens the door to coupling Blaze2D with higher-level photonic-design frameworks, including inverse design, perturbative envelope theories, and topological-invariant computations, in a way that treats the solver as a service rather than a monolith.
Within its stated scope of two-dimensional photonic crystals at moderate resolution, swept over many configurations, with full Bloch-level data exposed, Blaze2D matches MPB on accuracy, exceeds it on throughput in the mixed-precision regime, and provides a cleaner programming interface for downstream analyses. The solver is released under an open-source licence with the intent that other groups can re-use it as a drop-in component of their own pipelines.

References

[1] S. G. Johnson and J. D. Joannopoulos. "Block-iterative frequency-domain methods for Maxwell's equations in a planewave basis". In: *Optics Express* 8.3 (2001), pp. 173–190. DOI: 10.1364/OE.8.000173.

[2] R.-M. Lehner. *Repository: Blaze2D - A Rust-based 2D PWE Maxwell Solver*. Open-source software. MIT License. 2026. URL: https://github.com/RnLe/blaze2d.

[3] R.-M. Lehner. *PyPI: blaze2d*. Version 0.5.1. Python package on PyPI. 2026. URL: https://pypi.org/project/blaze2d/.

[4] A. Farjadpour, D. Roundy, A. Rodriguez, M. Ibanescu, P. Bermel, J. D. Joannopoulos, S. G. Johnson, and G. W. Burr. "Improving accuracy by subpixel smoothing in the finite-difference time domain". In: *Optics Letters* 31.20 (2006), pp. 2972–2974. DOI: 10.1364/OL.31.002972.

[5] J. Woo, S. Kim, and W. Y. Kim. "Dynamic Precision Approach for Accelerating Large-Scale Eigenvalue Solvers in Electronic Structure Calculations on Graphics Processing Units". In: *Journal of Chemical Theory and Computation* 19.5 (2023), pp. 1457–1465. DOI: 10.1021/acs.jctc.2c00983.

[6] A. V. Knyazev. "Toward the Optimal Preconditioned Eigensolver: Locally Optimal Block Preconditioned Conjugate Gradient Method". In: *SIAM Journal on Scientific Computing* 23.2 (2001), pp. 517–541. DOI: 10.1137/S1064827500366124.

[7] A. Stathopoulos and K. Wu. “A Block Orthogonalization Procedure with Constant Synchronization Requirements”. In: *SIAM Journal on Scientific Computing* 23.6 (2002), pp. 2165–2182. DOI: 10.1137/S1064827500370883.