Blaze2D
A 2D Maxwell Solver in Rust

Technical Report

Blaze2D: A High-Performance Solver for Photonic Crystals

Achieving order-of-magnitude speedups through a mixed-precision LOBPCG algorithm and cache-aware architecture.

Author:
Rene-Marcel Lehner

Repository:
github.com/RnLe/blaze2d
Package:
pypi.org/project/blaze2d

Published
January 8, 2026

Imprint

Project:	Blaze2D Technical Report
Title:	Blaze2D: A High-Performance Solver for Photonic Crystals
Author:	Rene-Marcel Lehner
Date:	January 8, 2026
Keywords:	photonic crystal, plane-wave expansion, LOBPCG, mixed precision, MPB, benchmarks, Rust

Repository: https://github.com/RnLe/blaze2d
Package: https://pypi.org/project/blaze2d/
Source: https://rnle.github.io/blaze2d/blaze/

Contents

1 Single-Core Performance 1
2 Multi-Core Performance 1
3 Memory Efficiency 2
3.1 Memory Scaling Laws 3
4 Resolution and Geometry 4
5 Accuracy Validation 5
6 Deviation Analysis 6
7 Parallel Scaling 6
8 Scaling with Resolution 7
9 Convergence and Iteration Count 9
10 Dielectric Contrast 10
11 Scaling with Band Count 10
12 Conclusion 11
References 11

Abstract

Photonic band structure calculations commonly rely on the Plane Wave Expansion (PWE) method, and MIT Photonic Bands (MPB) has long served as a widely used reference implementation for it [1,2]. MPB is trusted for its accuracy and is therefore a natural baseline for performance comparisons.

We introduce **Blaze**, a Rust-based solver that modernizes the PWE approach using mixed-precision arithmetic and an improved LOBPCG algorithm [3,4]. By explicitly targeting memory bandwidth bottlenecks, Blaze offers superior single- and multi-core scaling, and achieves a 95% reduction in memory footprint for regular use cases while maintaining reference accuracy.

1 Single-Core Performance

> Performance benchmarks utilize the canonical square and hexagonal lattice configurations from Joannopoulos' seminal 1997 Nature paper [5]. Unless otherwise specified, all data reflects the square lattice configuration at a resolution of 64, computing the lowest 8 bands with 20 k-points per segment between two high-symmetry points.

The computational cost of PWE solvers is dominated by Fast Fourier Transforms (FFTs) [1]. In this plane-wave formulation, Transverse Electric (TE) modes are more expensive to solve than Transverse Magnetic (TM) modes, because each operator application requires six FFTs rather than two for TM.

This complexity penalty is clearly visible in the legacy solver (Figure 1). Blaze, however, mitigates this through algorithmic optimizations. Even in Full Precision (`f64`), Blaze outperforms MPB. The decisive leap comes from the Mixed Precision (`f32/f64`) approach, which reduces memory traffic enough to effectively double the throughput, resulting in a total speedup of approximately 3×.

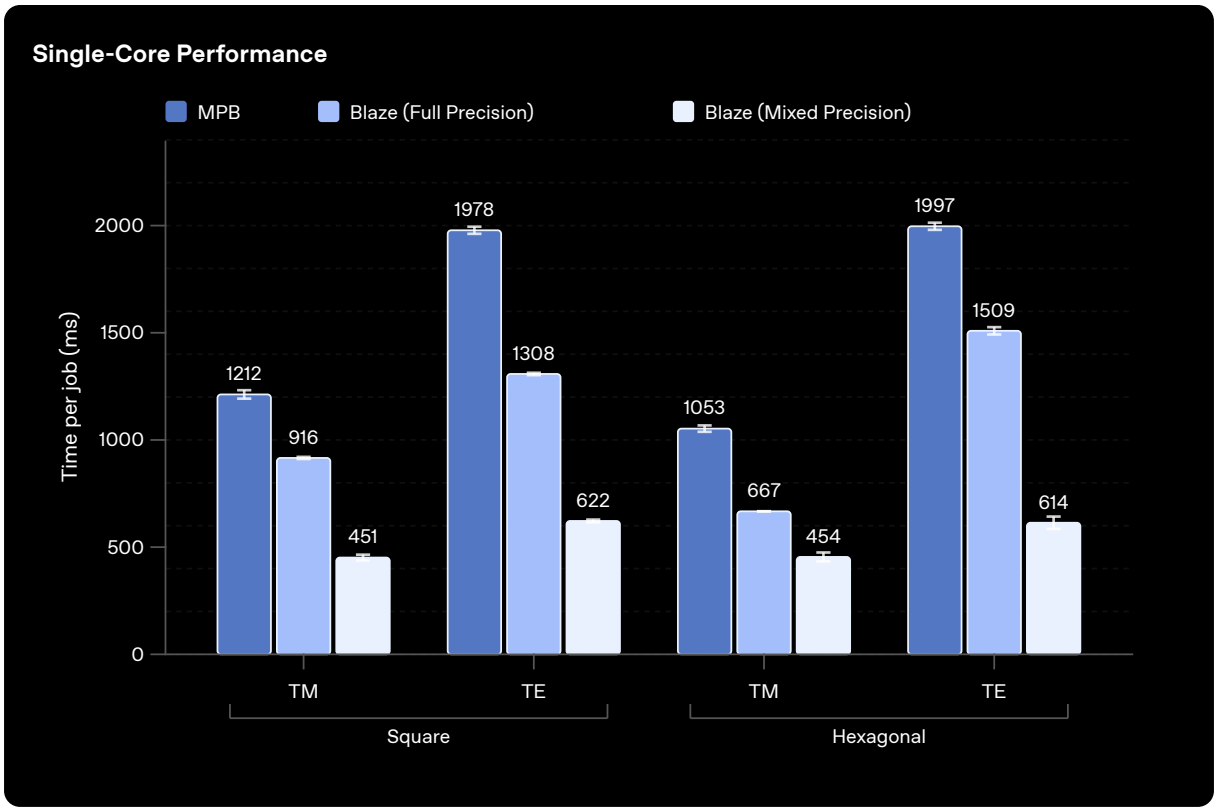

Figure 1: Blaze achieves 2.9× average speedup over MPB on single-core workloads. Mixed precision (`f32/f64`) shown alongside full precision (`f64`).

2 Multi-Core Performance

The architectural age of legacy solvers is most evident in parallel execution [6,7]. MPB attempts to parallelize individual operations ***within*** an iteration. On modern hardware, where small-to-medium lattice problems fit entirely within the CPU cache, this fine-grained threading introduces synchronization overhead that outweighs the computational gains, causing performance to regress as threads are added.[1]

[1]Figure 2.2 of [6] illustrates the dramatic divergence in performance trends between processor speed and memory bandwidth, showing how memory access has become the dominant bottleneck in modern computing systems.

Blaze avoids this overhead by parallelizing entire jobs rather than individual operations, a strategy optimized for large-scale parameter sweeps.

Figure 2 reveals a clear three-tier performance hierarchy. The legacy solver struggles with thread overhead. Blaze in Full Precision scales efficiently but remains sensitive to the higher algebraic load of TE modes (approx. 240–350 ms). In contrast, the Mixed Precision mode hits a "hard floor" at roughly 160 ms. By halving the memory requirement for state vectors, Blaze masks much of the computational complexity of the TE mode, consistent with a solver that has entered a memory-bandwidth-limited regime on a given machine [7].

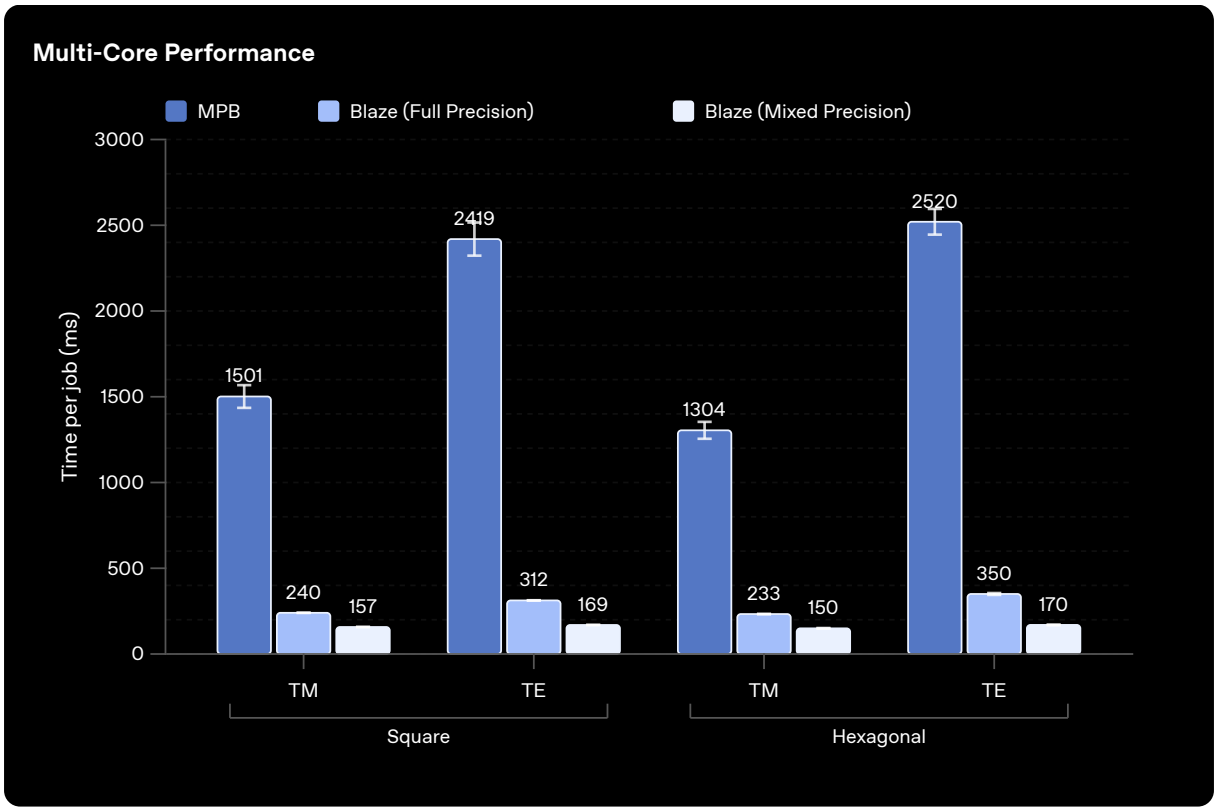

Figure 2: Blaze achieves 11.9× average speedup over MPB on 16-thread workloads. Mixed precision (`f32/f64`) shown alongside full precision (`f64`).

> All subsequent benchmarks for Blaze are performed in Mixed Precision mode, unless otherwise specified.

3 Memory Efficiency

High-performance computing is increasingly defined by data movement [7]. A major limitation of MPB is its static memory management; benchmarks reveal that the legacy solver reserves a large, fixed memory block (approx. 190 MB) ***regardless*** of the problem size.

Blaze adopts a dynamic allocation strategy. As Figure 3 shows, this results in a dramatic reduction in peak memory usage for standard resolutions. This reduction is critical: by keeping the working set small, Blaze allows the CPU to operate almost entirely within its high-speed L3 cache, avoiding the latency penalty of fetching data from main RAM.

Fundamentally, the storage requirements for FFTs and operator workspaces scale directly with the grid resolution (N). Therefore, analyzing memory growth against resolution provides the most critical insight into the architectural efficiency.

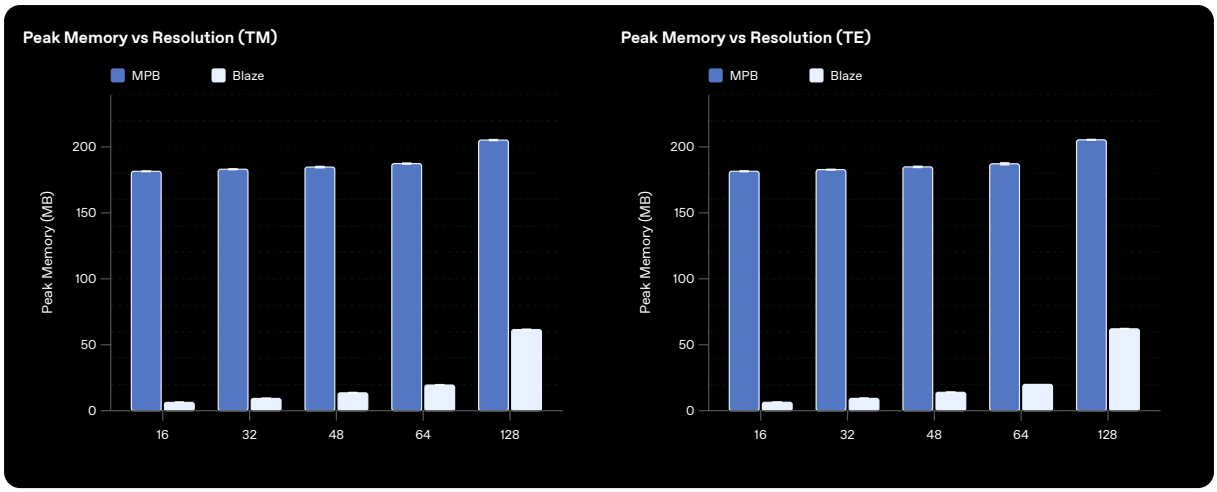

Figure 3: Peak memory against grid resolution, for both polarizations. MPB holds a fixed footprint of roughly 190 MB at every resolution, while Blaze allocates what the problem actually requires.

This efficiency extends to the dimensionality of the search space. In the LOBPCG algorithm, the search space size is determined by the number of bands ($3n$) [3]. While one might expect memory usage to scale with this complexity, both solvers maintain a constant footprint even as the number of bands increases (Figure 4).

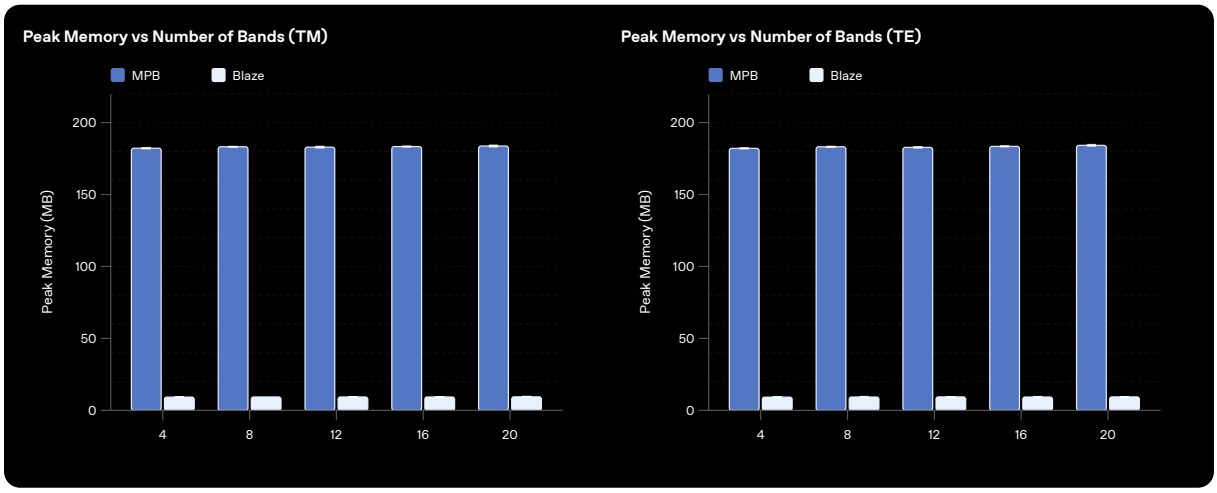

Figure 4: Peak memory against the number of requested bands. Neither solver's footprint grows with the size of the LOBPCG search block.

3.1 Memory Scaling Laws

To understand the limits of this efficiency, we analyzed how the relative advantage evolves (Figure 5). At low resolutions, MPB is dominated by its static overhead, giving Blaze a 20× advantage. As the resolution increases, the physical storage requirements for the grid naturally grow, and the ratio asymptotically approaches 1×. As mentioned, for the number of bands sweep, both solvers maintain constant memory usage, resulting in a flat ratio.

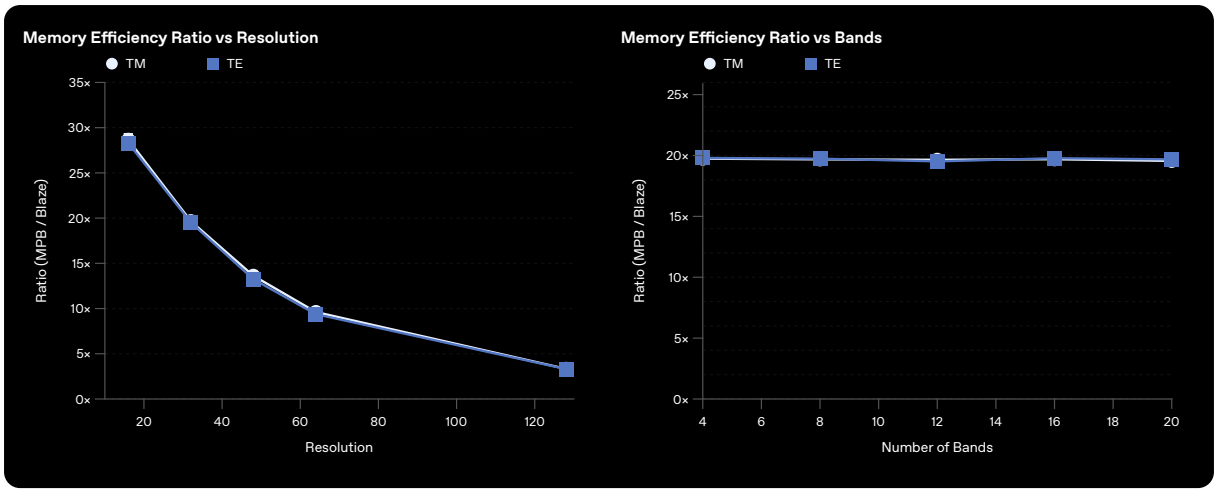

Figure 5: Ratio of MPB's peak memory to Blaze's, swept over resolution (left) and band count (right). The advantage is largest where MPB's fixed allocation dominates, and decays as the grid itself becomes the cost.

For varying resolutions, MPB's memory usage is effectively constant ($N^{0.06}$), confirming the pre-allocation hypothesis. In contrast, Blaze follows a near-linear trend ($N^{1.09}$), scaling predictably with the problem size. Notably, this footprint is identical for both TM and TE polarizations, proving that the storage cost in Blaze is determined strictly by grid topology, independent of the operator's computational complexity.

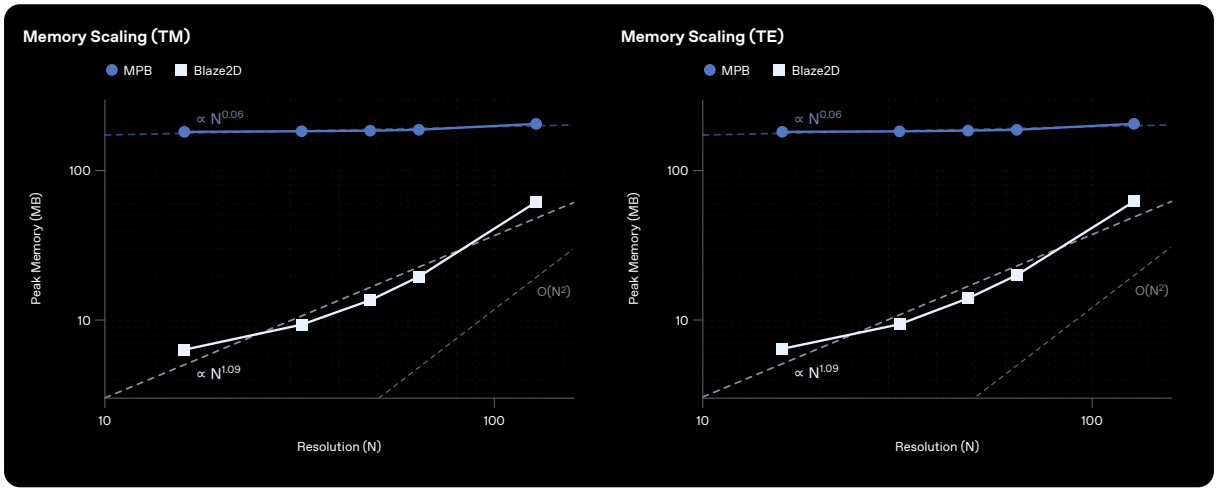

Figure 6: Peak memory against resolution on log-log axes, with power-law fits. MPB is flat at $N^{0.06}$; Blaze follows $N^{1.09}$, well below the $O(N^2)$ reference.

4 Resolution and Geometry

Several of the benchmarks below vary the grid resolution, measured in pixels per unit cell. Figure 7 shows what a given resolution looks like for a circular rod, which helps build intuition for how coarse these grids actually are. It also shows the subpixel smoothing applied at the rod boundary: instead of sampling the permittivity on a hard pixel grid, which would produce jagged staircase edges, the dielectric is smoothed at the corners. Blaze uses the same subpixel smoothing method as MPB [8]. In practice, resolutions of 32 to 64 pixels per unit cell are enough for research-grade band structures, and higher values are rarely necessary unless the unit cell contains many atoms.

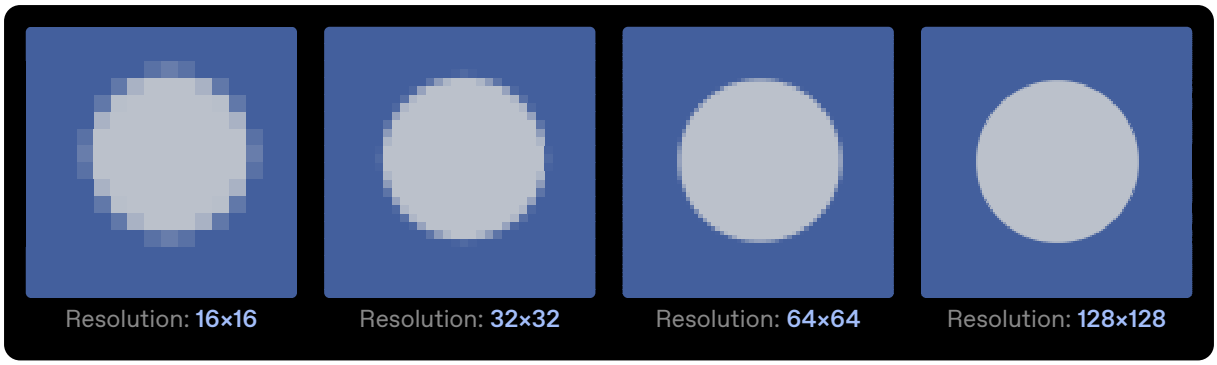

Figure 7: Smoothed permittivity map of a circular rod at four grid resolutions. The boundary cells take intermediate values from subpixel smoothing rather than snapping to $\varepsilon = 1$ or $\varepsilon = 13$, which is what keeps a coarse grid usable. On the website this figure is an interactive slider.

5 Accuracy Validation

The reduction in precision and memory footprint raises a direct question: do we gain speed at the cost of accuracy? To answer this question, we compared the eigenfrequencies from Blaze against a high-precision MPB reference along the full $\Gamma \to X \to M \to \Gamma$ path. In Figure 8, the MPB bands are drawn as lines and the Blaze eigenvalues as markers.

A short note on methodology. MPB tracks each band *adiabatically* across avoided crossings, while Blaze reports the lowest eigenvalues at each $\boldsymbol{k}$-point without assigning band identities. The two conventions can disagree on the highest band of the computed set, where a crossing may exchange a band with the next one just outside the set. To keep the comparison clean we compute the lowest 20 bands with both solvers but show only the lowest 10, and for Blaze we plot the lowest eigenvalues directly with no band matching. Every displayed band is then well inside the computed window, so both solvers report the same set of values and the band-tracking difference does not appear here.

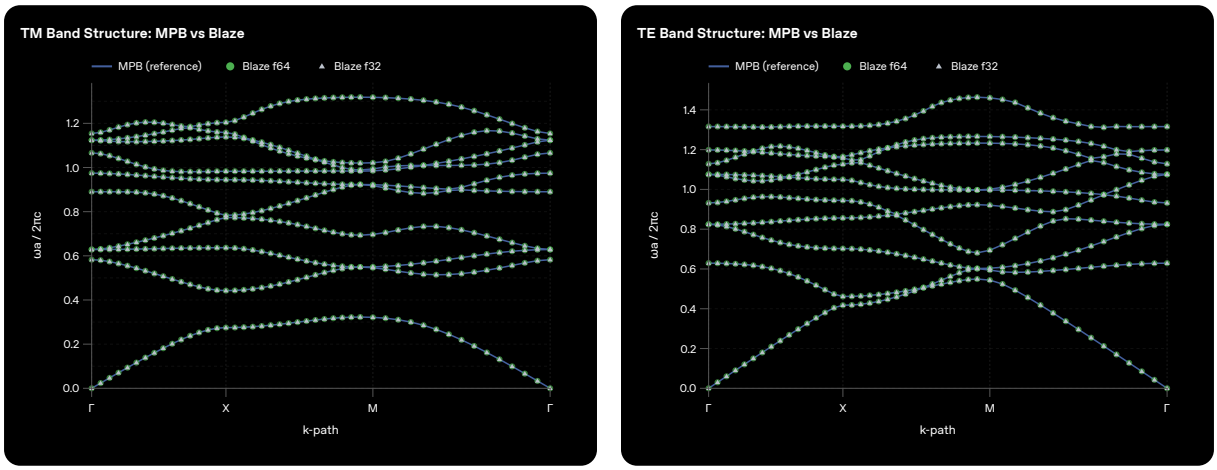

Figure 8: Band structures along $\Gamma \to X \to M \to \Gamma$ for TM (left) and TE (right). MPB is drawn as lines, Blaze in full and mixed precision as markers. The three curves are indistinguishable at this scale.

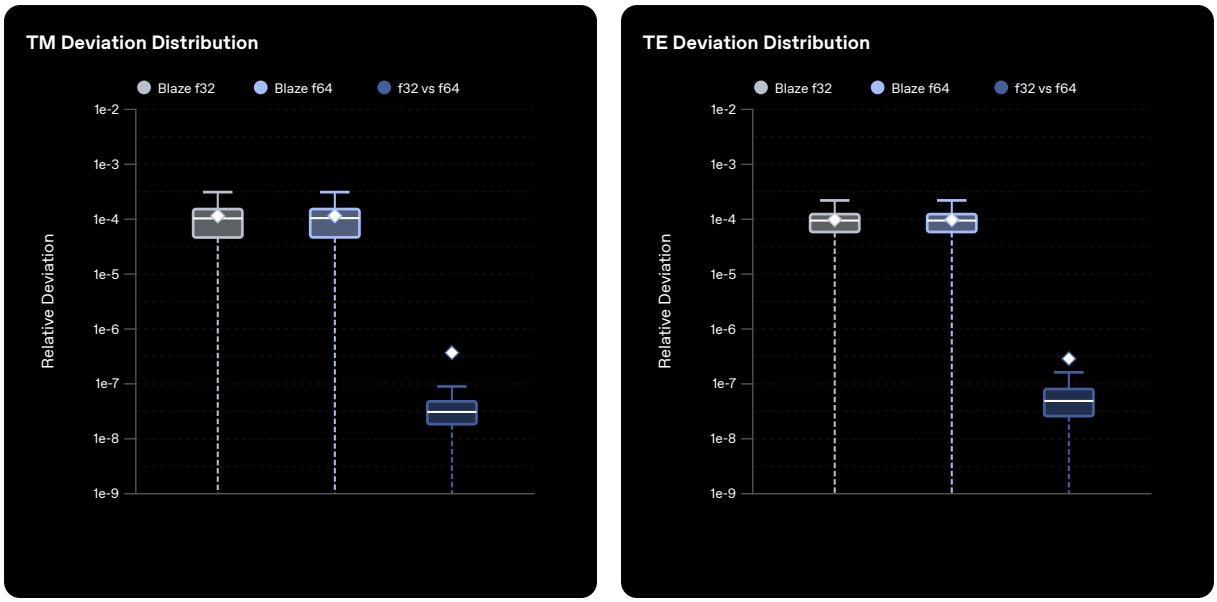

Figure 9: Distribution of the relative deviation across all bands and $\boldsymbol{k}$-points, for TM (left) and TE (right). The first two boxes measure each Blaze run against MPB; the third measures the two Blaze runs against each other.

6 Deviation Analysis

Across the full path, both polarizations agree with the MPB reference to within Blaze's convergence tolerance (Figure 9). The relative deviation has a median of about 1×10^{-4}, with a worst case of 4×10^{-4} for TM and 3×10^{-4} for TE. The agreement does not degrade with band index; it is flat from the lowest band to the tenth.

The comparison between the `f32` and `f64` runs of Blaze is just as informative: they differ by a median of about 1×10^{-7} and a maximum below 5×10^{-5}, more than an order of magnitude smaller than either run's deviation from MPB. The eigenvalues from mixed precision and full precision are, in effect, identical. That outcome is consistent with prior mixed-precision LOBPCG results in electronic-structure calculations, where single-precision storage with double-precision critical reductions preserved eigenvalue accuracy while accelerating solves [4]. Here, the higher noise floor of single-precision field storage remains below the convergence floor of the runs, so the 3× speedup and the 95% memory reduction come at no measurable cost in accuracy.

7 Parallel Scaling

The two solvers parallelize in different ways. MPB splits the work inside a single solve across threads, while Blaze runs many independent solves at the same time, one per thread, which suits the parameter sweeps it is built for. Figure 10 sweeps the thread count from 1 to 16 and measures throughput in solves per second, for a small problem ($N = 16$) and a large one ($N = 128$).

For the small problem, Blaze scales almost linearly, from 28 to 220 solves per second across 16 threads. MPB's threaded mode does not benefit: it stays near 14 solves per second and even drops slightly, because the cost of coordinating threads within a single solve outweighs the gain. Running MPB as separate processes, one solve per core, does scale, but reaches only about half of Blaze's throughput.

For the large problem the absolute numbers are smaller for every solver, but the pattern is the same. Blaze scales by about 3× across the sweep (Figure 11), while MPB's threaded mode stays flat. The practical conclusion is that **parallelizing across independent jobs works better than parallelizing within a single solve**.

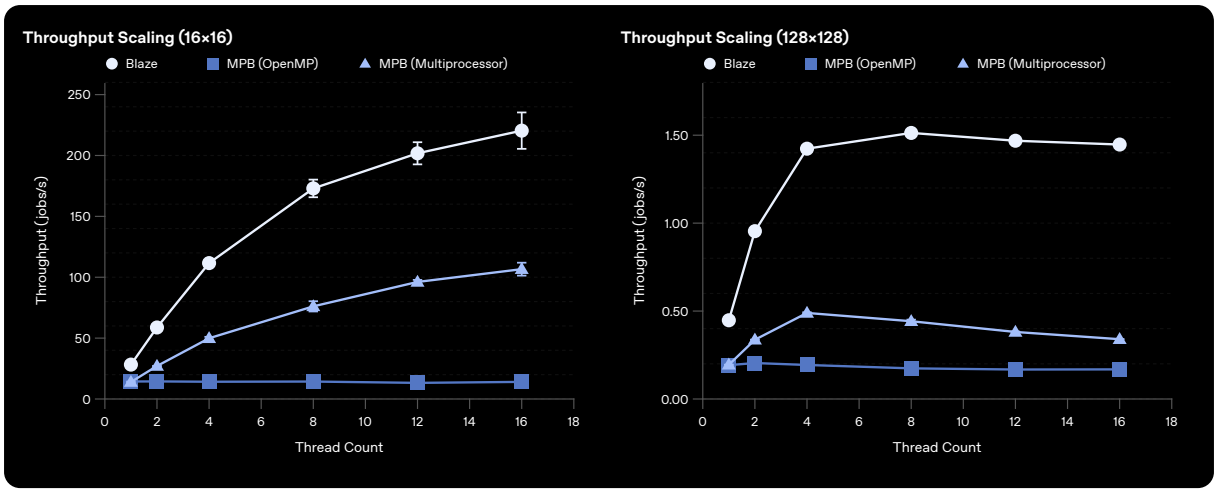

Figure 10: Throughput in solves per second against thread count, for a small problem ($N = 16$, left) and a large one ($N = 128$, right).

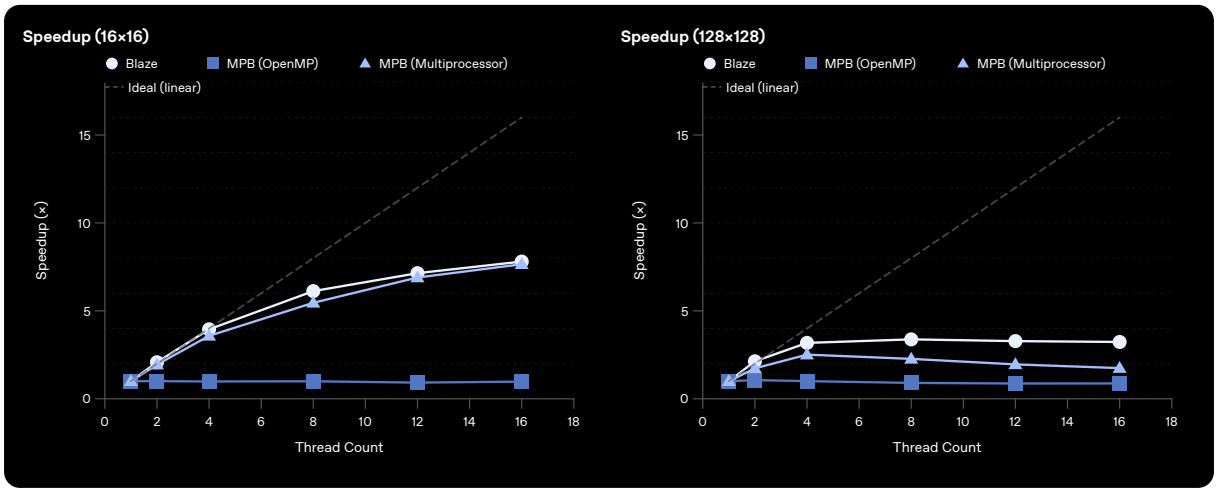

Figure 11: The same sweep expressed as speedup over the single-threaded result. Blaze tracks the ideal line closely at $N = 16$; MPB's threaded mode is flat in both panels.

8 Scaling with Resolution

Resolution sets the overall cost: the number of plane waves grows as N^2 and the FFTs as $N^2 \log N$. Figure 12 sweeps the grid from $N = 16$ to $N = 192$ and compares wall-clock time against MPB.

Blaze is faster at every resolution, and its solve time grows smoothly and predictably with N. MPB behaves differently. Its timings are uneven across resolutions, performing noticeably better at powers of two (16, 32, 64, 128) and jumping in cost at the values in between. At $N = 96$ MPB is already almost as slow as at $N = 128$, and the step up to $N = 192$ is large. The exact cause is not certain, but it most likely comes down to how the FFT plans and operator tiling handle sizes that are not powers of two. Figure 13 shows the combined effect: Blaze's lead grows toward higher N, reaching roughly 5× at $N = 192$ (8.0 s against 39 s for TM, 12 s against 70 s for TE).

The log-log scaling plot in Figure 14 makes the trends comparable. Blaze follows a clean power law, while MPB's curve is jumpy and trends toward a steeper slope at high resolution.

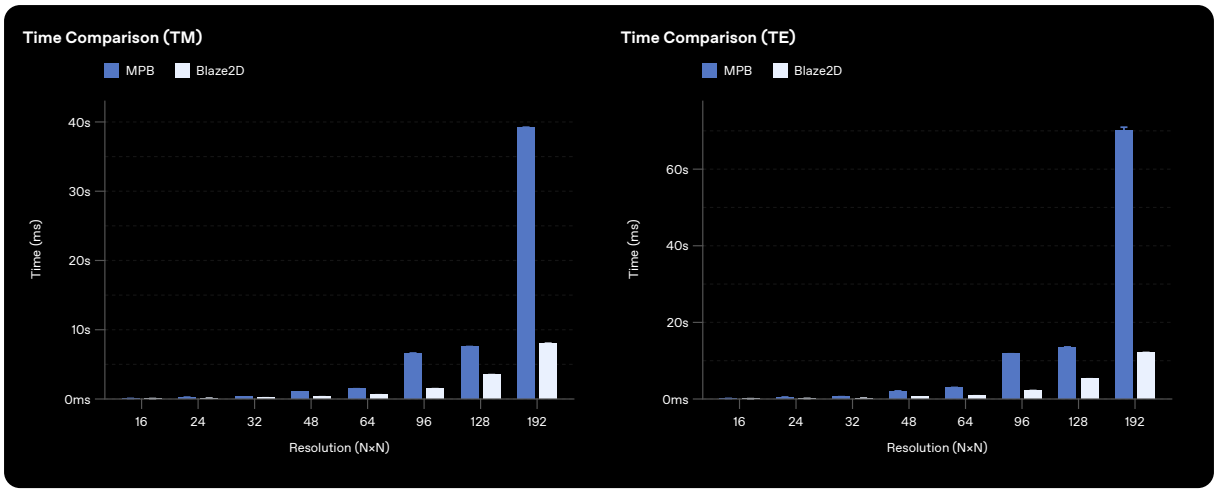

Figure 12: Wall-clock solve time against grid resolution for TM (left) and TE (right). MPB's cost is uneven across resolutions; Blaze's grows smoothly.

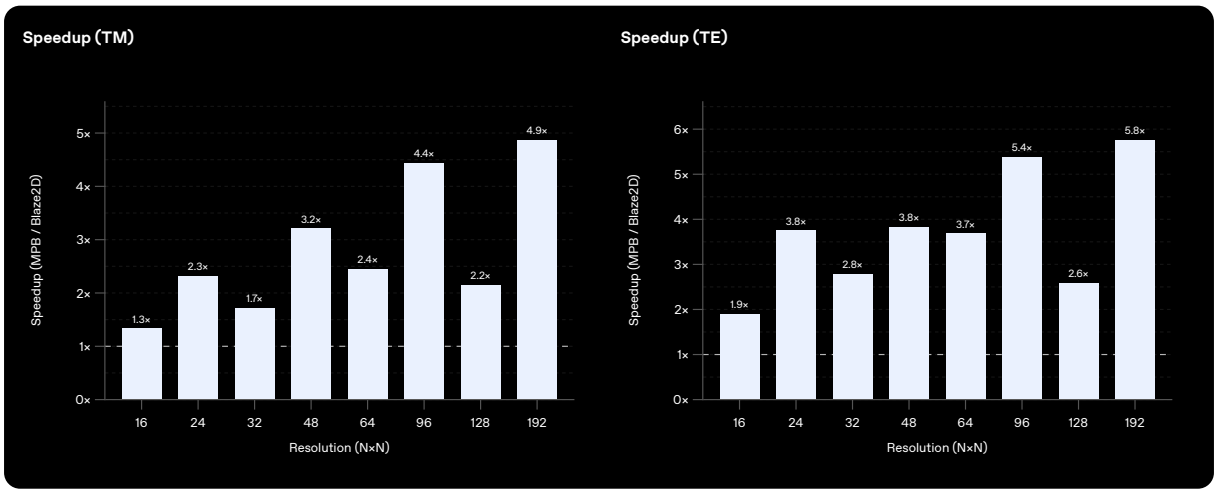

Figure 13: Blaze's speedup over MPB across the same sweep. The lead widens with resolution rather than saturating.

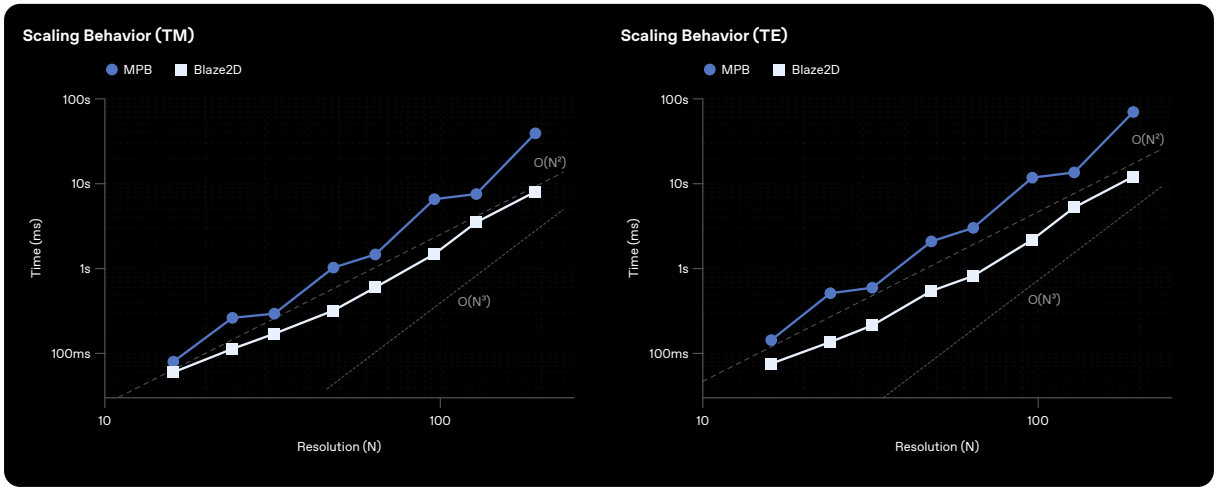

Figure 14: The same data on log-log axes with power-law fits, which makes the two solvers' scaling exponents directly comparable.

9 Convergence and Iteration Count

Beyond the cost of the *entire* single iteration, the number of iterations needed to converge each $\boldsymbol{k}$-point matters.

Blaze converges in fewer iterations than MPB on average, about 4.5 per $\boldsymbol{k}$-point for TM and 5.6 for TE, against 7.5 and 11.9 for MPB (Figure 15). Both solvers **warm-start** each $\boldsymbol{k}$-point from the converged solution of the previous one (Blaze adopts this technique from MPB), so the warm start cannot explain the difference. The most likely explanation is a combination of soft-locking, where converged bands are kept in the working space but no longer refined, and the different convergence criteria the two solvers use: Blaze stops once the eigenvalues have stabilized, while MPB monitors the residual. There may also be a general efficiency difference in the LOBPCG variant.

MPB's iteration count spikes at certain $\boldsymbol{k}$-points, most often around high-symmetry points, where it reaches 39 iterations for TM and 85 for TE. Blaze has occasional spikes as well, but they are smaller (up to 12 and 24) and tend to fall at *different* $\boldsymbol{k}$-points (Figure 17). That the spikes occur in different places is consistent with the two solvers using different stopping criteria, which become sensitive under different conditions rather than at the same points. The polarization gap is visible throughout: TM, whose operator is diagonal and easy to precondition, converges in fewer iterations than TE for both solvers.

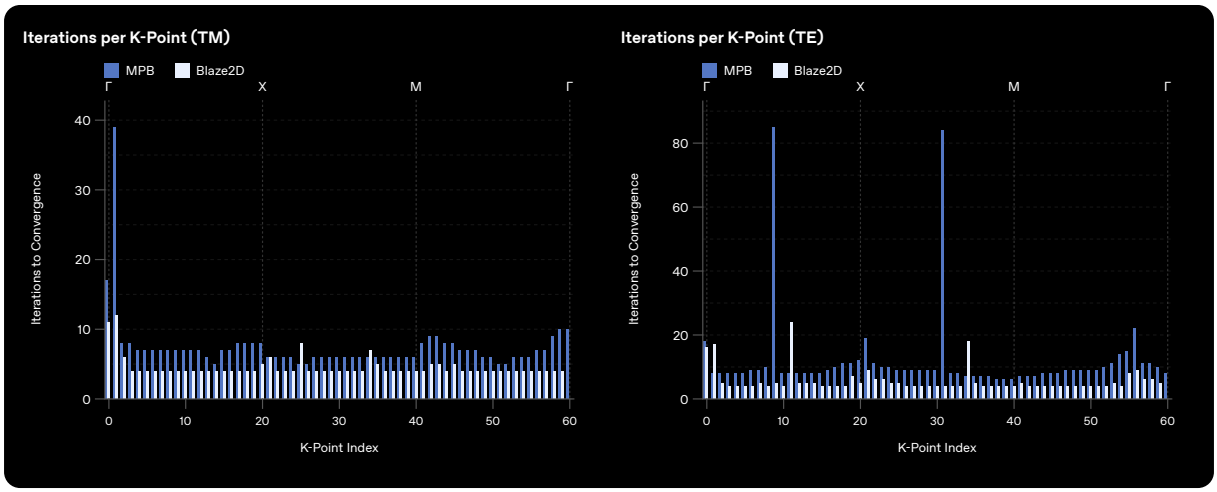

Figure 15: Mean iteration count per $\boldsymbol{k}$-point for TM (left) and TE (right).

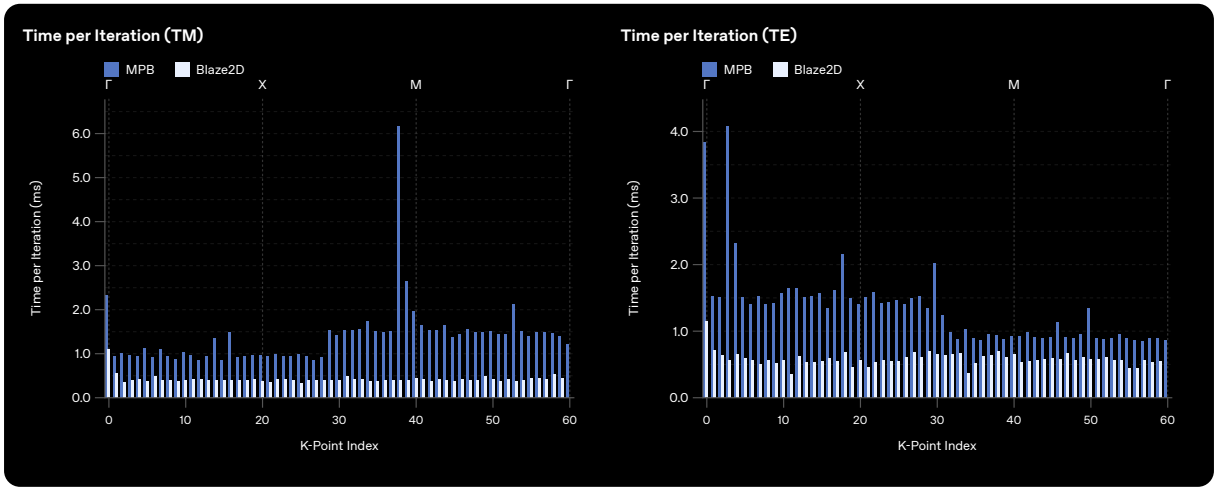

Figure 16: Time per iteration for the same runs. Blaze's per-iteration cost is lower as well, so the iteration-count advantage compounds.

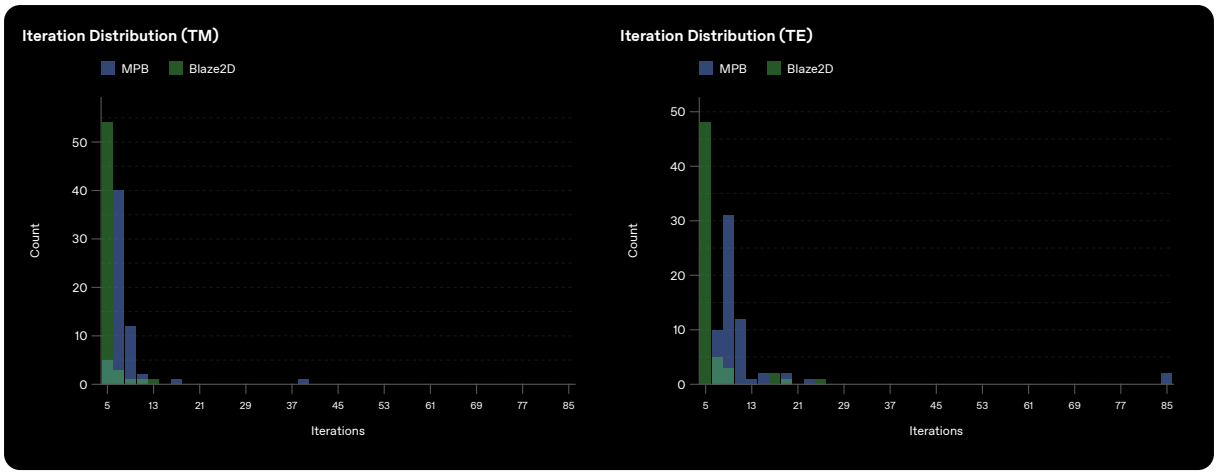

Figure 17: Distribution of iteration counts over all k-points. MPB's tail reaches 39 iterations for TM and 85 for TE; Blaze's stays short.

10 Dielectric Contrast

The benchmark in Figure 18 varies the rod permittivity from $\varepsilon = 2$ to $\varepsilon = 13$, covering the range where photonic band gaps open. Blaze's solve time stays essentially flat across the whole range, which is a useful stability property: the cost does not depend on the contrast of the crystal. MPB's behavior is more uneven. Its timings vary substantially from one contrast to the next, with occasional sudden drops and jumps. MPB is therefore more sensitive to the dielectric contrast than Blaze.

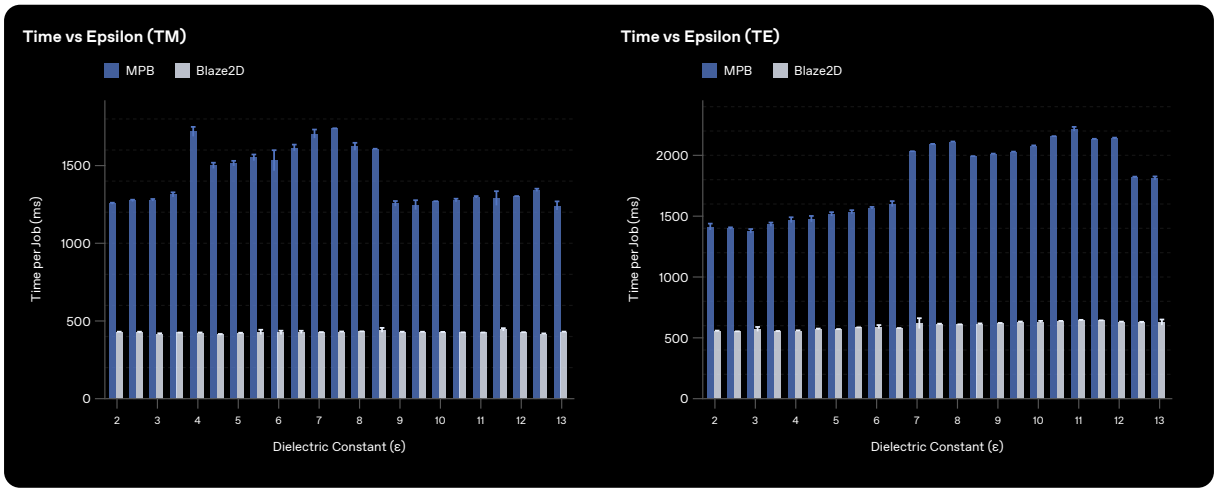

Figure 18: Solve time against rod permittivity for TM (left) and TE (right). Blaze is flat across the sweep; MPB is not.

11 Scaling with Band Count

The sweep in Figure 19 varies the number of bands requested, from 4 to 20. The cost grows steadily with band count for both solvers, which is expected: more bands mean a larger LOBPCG search block and more vectors to orthogonalize each iteration. Blaze keeps a roughly 2× lead across the whole range, solving the 20-band TM problem in 1.9 s against MPB's 3.9 s. As Section 3 showed, this added work does not increase the memory footprint, so requesting more bands costs time but not space.

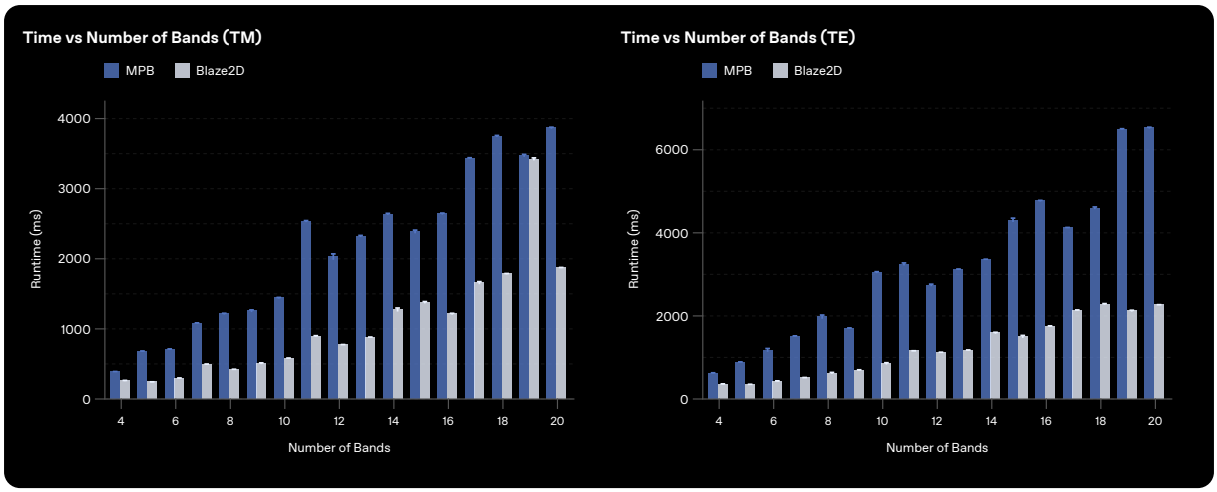

Figure 19: Solve time against the number of requested bands for TM (left) and TE (right).

12 Conclusion

Blaze was built to solve 2D photonic crystal band structures quickly, repeatedly, and without surprises. The benchmarks underline this behavior. In mixed precision Blaze is roughly 3× faster than MPB on a single core, scales almost linearly across threads where MPB's threaded mode stalls, and uses up to 20× less memory at the resolutions used in practice. It does so while reproducing MPB's eigenvalues to within $\sim 10^{-4}$ (its own convergence tolerance) with single and double precision giving indistinguishable results. Its cost is also predictable: solve time grows smoothly with resolution and stays flat across dielectric contrast, in both of which MPB is uneven.

None of this is accidental. It follows from starting on a modern foundation: a memory-aware, mixed-precision LOBPCG core written in Rust, parallelized across whole solves rather than inside them, and shipped as an installable Python package. Performance, scalability, and simplicity were design goals from the start.

The most consequential advantage does not appear in any timing plot. Because Blaze is a plane-wave solver that exposes its internals, it returns far more than frequencies: the Bloch functions themselves, and the operator matrix elements assembled from them, such as Berry connections and effective-mass tensors [9–11]. Standard band solvers like MPB do not surface this data. This allows the user to look inside the operator instead of only at its spectrum, and to do so for thousands of crystal configurations in a single sweep. For routine band-structure work the speed is the headline. For the physics that follows, the ability to extract this interior structure at scale is what turns a fast solver into a foundation to build on.

References

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

[2] K. Sakoda, *Optical Properties of Photonic Crystals*, 2nd ed., Vol. 80 (Springer, Berlin, 2005).

[3] A. V. Knyazev, Toward the Optimal Preconditioned Eigensolver: Locally Optimal Block Preconditioned Conjugate Gradient Method, SIAM Journal on Scientific Computing **23**, 517 (2001).

[4] 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, Journal of Chemical Theory and Computation **19**, 1457 (2023).

[5] J. D. Joannopoulos, P. R. Villeneuve, and S. Fan, Photonic crystals: putting a new twist on light, Nature **386**, 143 (1997).

[6] J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach*, 5th ed. (Morgan Kaufmann, Waltham, MA, 2011).

[7] S. Williams, A. Waterman, and D. Patterson, Roofline: An Insightful Visual Performance Model for Multicore Architectures, Communications of the ACM **52**, 65 (2009).

[8] 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, Optics Letters **31**, 2972 (2006).

[9] M. V. Berry, Quantal Phase Factors Accompanying Adiabatic Changes, Proceedings of the Royal Society of London. Series A **392**, 45 (1984).

[10] F. Wilczek and A. Zee, Appearance of Gauge Structure in Simple Dynamical Systems, Physical Review Letters **52**, 2111 (1984).

[11] P.-O. Löwdin, A Note on the Quantum-Mechanical Perturbation Theory, The Journal of Chemical Physics **19**, 1396 (1951).