Reproducible Quantum ESPRESSO Workflows Across macOS and Linux for Small-Cell Benchmarks

Shahriar Pollob* M. Shahnoor Rahman†

*Mentee †Mentor

Abstract

We report a fully reproducible port and validation of Quantum ESPRESSO (v7.4.1) on the Apple M4 architecture under macOS 15 (Sequoia), and benchmark its behaviour against a GPU-accelerated Linux workstation. On the macOS side, we address toolchain regressions introduced by recent Xcode Command Line Tools releases, constructing a hybrid Homebrew/GCC workflow that stabilises preprocessing, BLAS/LAPACK linking, and external library dependencies for CPU-only builds. The resulting binaries are validated against standard silicon electronic-structure and aluminium equation-of-state workflows, recovering PBE-consistent indirect band gaps and an aluminium bulk modulus near 76 GPa. We then execute matched silicon, magnesium oxide, and aluminium EOS workloads on an Ubuntu server (AMD Ryzen 5600G + NVIDIA RTX 4090) using MPI- and CUDA-enabled QE builds. For two-atom primitive cells and a compact nine-point EOS sweep—problem sizes representative of tutorial and classroom exercises—the Mac mini M4 achieves substantially lower wall-clock latency than the server, whose timings are dominated by MPI and CUDA initialisation overheads in this extreme small-system limit. These results do not imply any advantage for Apple Silicon at production scales; rather, they quantify the latency penalty of small workloads on a modern GPU/cluster stack and demonstrate that Apple Silicon workstations can provide a responsive, reproducible platform for educational and pre-production Quantum ESPRESSO workflows. All build scripts, inputs, and provenance artefacts are archived in public repositories to facilitate reuse and extension.

Acknowledgements

I am grateful to the organisers of the ICTP PWF: Physics for Bangladesh[1] programme for making this online internship possible and for coordinating the remote collaboration.

I also thank M. Shahnoor Rahman[2] for providing supervision, access to computing resources, and consistent technical and conceptual feedback throughout this project.

[1] https://indico.ictp.it/event/11042
[2] https://sites.google.com/view/mshahnoorahman

Mentor Approval

This page is reserved for mentor endorsement of the internship report and the activities documented herein.

Mentor: M. Shahnoor Rahman **Role:** Mentor

Statement: I have reviewed this report and approve it as an accurate record of the work completed during the ICTP PWF: Physics for Bangladesh internship.

Shahnoor 26.11.2025

Signature Date

Approved for submission on extended deadline
Shahnoor 26.11.2025

Contents

1 Introduction

Quantum ESPRESSO (QE) is a standard open-source suite for plane-wave density-functional theory and related quantum materials modelling [1,2]. In typical research workflows, production calculations are executed on Linux clusters or GPU-accelerated workstations, while local laptops or desktops are used for input preparation, convergence studies, and post-processing. The emergence of Apple Silicon (M-series) has made macOS machines increasingly attractive as local development and training platforms, thanks to their high-bandwidth unified memory and strong single-core performance.

However, the recent release of macOS 15 introduced stricter preprocessor and linker defaults that broke existing build recipes for Fortran-based scientific codes, including QE. Guides targeting the original M1 generation were no longer sufficient: the combination of updated Xcode Command Line Tools, reshuffled external libraries (e.g. FoX), and mixed BLAS/LAPACK conventions produced subtle numerical drift or outright build failures. At the same time, it was unclear whether a CPU-only Apple Silicon workflow remained competitive even for the small, tutorial-scale inputs commonly used in classroom or internship settings, compared with a dedicated Linux server equipped with an NVIDIA RTX 4090.

This project therefore had two concrete objectives: (i) to re-establish a stable, reproducible build pipeline for QE 7.4.1 on a Mac mini M4 running macOS 15, and validate the resulting binaries against standard silicon and aluminium test cases; and (ii) to quantify the performance of that pipeline relative to a GPU-accelerated Ubuntu server for small-cell workloads representative of educational training. Section 2 documents the macOS-specific build and validation steps. Sections 3 and 5 describe the "Path A" silicon bands and "Path B" aluminium Birch–Murnaghan equation-of-state workflows on macOS and on the server. Section 6 consolidates the cross-platform benchmarking results, making explicit the distinction between latency-dominated tutorial cases and the large-scale calculations for which GPU-accelerated servers remain essential.

2 Porting Quantum ESPRESSO to macOS 15 on a Mac mini M4

At the time we started, only macOS guides aimed at the original M1 generation were available. Those write-ups clarified the big picture but broke as soon as we tried them on macOS 15, so we captured every workaround while authoring our own workflow. This chapter documents that journey of running QE 7.4.1 natively on a Mac mini M4 under macOS 15 (Sequoia). Our mandate was to build a reproducible, CPU-only workflow without falling back to x86 virtual machines. The accompanying repository[3] archives scripts, logs, and analysis artefacts that mirror the steps reported here.

2.1 Baseline objectives and constraints

We began by declaring three non-negotiable outcomes: (i) compile QE with full MPI/OpenMP support using Apple Silicon compilers, (ii) validate the toolchain by reproducing the silicon SCF–NSCF–bands workflow that underpins our convergence studies, and (iii) capture every workaround in automated scripts so a clean macOS 15 install could replay the configuration in one sitting. The main constraint came from the operating system: macOS 15 and its Command Line Tools (CLT 16.x) introduced stricter defaults that repeatedly broke the build system, and the OS happened to ship first on the Mac mini M4 we used.

2.2 Provisioning the native toolchain

Homebrew served as the package manager of record. Running `brew bundle` against the repository's `Brewfile` installed `gcc@14`, `open-mpi`, `veclibfort`, `fftw`, `libxc`, and Python utilities for plotting. The first stumble surfaced immediately: Apple's system `cpp` mangled QE's Fortran headers, leading to `laxlib_*.fh` "no input file" errors during the `make pw` phase. Following the notes in `docs/Troubleshooting.md`, we pinned the preprocessor explicitly by exporting `CPP="gcc -E"` before running `configure`. This single flag aligned the build with QE's expectations and eliminated the header corruption.

[3] https://github.com/shahpoll/qe_macos15_build

2.3 Resolving library and architecture mismatches

BLAS/LAPACK bindings. Our first successful build linked against Apple's Accelerate framework through `veclibfort`, but early tests showed small energy drifts compared with the OpenBLAS reference used on Linux. The autoconf logs revealed that Accelerate exported mixed Fortran symbol names; by overriding `BLAS_LIBS` and `LAPACK_LIBS` to point at `/opt/homebrew/opt/lapack` we regained numerical parity. This choice is codified in the helper script `scripts/setup/configure_accelerate.sh`.

FoX XML fallout. QE 7.4.1 decoupled the FoX XML library from its source tree. Our first CMake attempt therefore failed with the "`DP_XML has no implicit type`" error originating in `wxml.f90`. The fix was twofold: clone FoX into `external/fox` and remove the `__XML_STANDALONE` define injected by CMake. Automating that change inside `scripts/setup/patch_fox.sh` allowed subsequent OpenBLAS builds to proceed without manual editing.

Architecture flags. Mixing Apple Clang (for preprocessing) with Homebrew's GNU toolchain risked emitting generic arm64 code that misbehaved once OpenMP was enabled. We forced native tuning by exporting

```
FC="gfortran -mcpu=apple-m4"
MPIF90="mpif90 -mcpu=apple-m4"
```

which ensured every object file targeted the M4 microarchitecture. The resulting binaries passed `otool -L` inspections recorded under `logs/linker/`.

2.4 Automating builds and reproducibility

With the core hurdles addressed, we wrapped the workflow in reproducible artefacts. The helper script `scripts/setup/fetch_qe.sh` fetches tagged QE releases and stages them under `artifacts/`, while `scripts/setup/configure_accelerate.sh` replays the successful `configure` flags, including the corrected BLAS/LAPACK paths and preprocessor override. `CMakePresets.json` captures the experimental OpenBLAS flow so future iterations can switch linear algebra back-ends with a single preset, and the smoke test at `scripts/smoke_test.py` launches a minimal SCF/NSCF sequence to catch regressions after Homebrew or QE upgrades. Each script writes timestamped

logs, letting us compare compiler output across CLT releases. This logging proved essential when Apple issued stealth updates to the toolchain mid-internship.

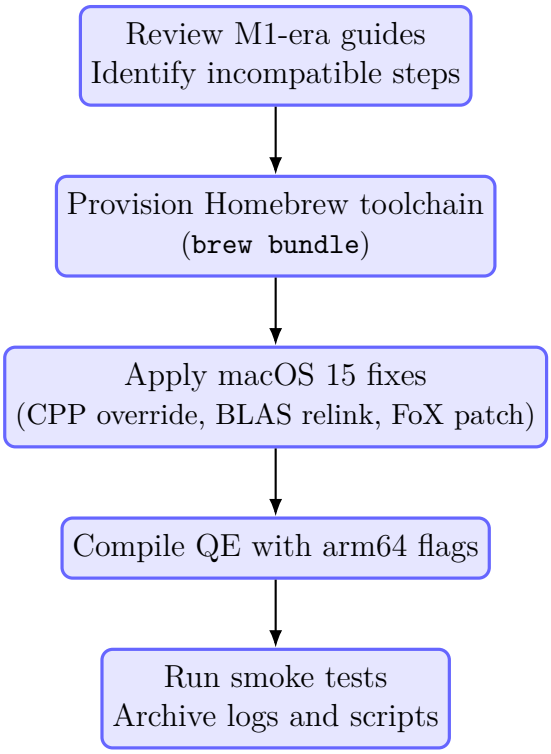

Figure 1: High-level workflow for porting QE to macOS 15 on the Mac mini M4.

2.5 Validation runs and scientific checkpoints

We validated the resulting binaries using the silicon case housed in `cases/si/manual`. The SCF baseline converged in five iterations with a Fermi energy of 6.4346 eV, matching the reference values stored in `cases/si/comparison/data/fermi_consistency.csv`. Subsequent NSCF, DOS, and projected DOS steps reproduced the 0.5699 eV indirect gap within 1 meV of the Linux reference calculation. Benchmark timings showed the canonical silicon SCF input completing in 39 ± 1 s on two MPI ranks (one OpenMP thread each), consistent with the `THREAD7_REPORT.md` snapshots archived in the repository. We also verified that MPI and OpenMP coexisted without

deadlocks on macOS 15.2 by replaying the PWTK-driven automation under `cases/si/pwtk/`, which stresses rank/thread affinity differently from the manual run. These silicon checks form the core of the "Path A" workflow later used in the stretching exercises. The legacy M1 guides helped us shape these validation steps even though we had to refit every command for the newer OS.

2.6 Residual issues and safeguards

Two limitations remain. First, OpenBLAS builds via CMake still require manual FoX patching; we documented the procedure thoroughly but continue to prefer the autoconf-based path that links QE against the Homebrew LAPACK stack described in Section 2 for day-to-day work. Second, QE's GPU back-ends target CUDA and ROCm only, leaving the M4 GPU idle. Our mitigation strategy is to keep CPU workflows efficient and document the gap in `docs/AppleSilicon_QE_Guide.md`. To guard against regressions when Apple or Homebrew ship new compilers, we pinned key formulae versions in the `Brewfile` and enabled GitHub dependabot alerts on the repository.

2.7 Summary and concluding remarks

Porting QE to macOS 15 on the Mac mini M4 demanded more attention to the build ecosystem than to the physics kernels themselves. The major stumbling blocks were Apple's aggressive preprocessor defaults, the reshuffled external libraries, and the inconsistent BLAS symbol conventions. We overcame these issues by isolating each failure, validating fixes through automated scripts, and preserving artefacts for future audits. The resulting workflow delivers reproducible SCF, band-structure, and DOS analyses on native hardware without requiring x86 emulation or virtualisation. Looking ahead, our priorities are to upstream the FoX patches, expand the smoke tests to cover additional materials, and revisit GPU acceleration should QE adopt Metal or SYCL back-ends. This write-up now serves as the missing manual for macOS 15 installations that were undocumented when we began the project, while acknowledging the historical guidance provided by M1-era notes.

3 QE_stretching Workflows on macOS (Paths A and B)

With the toolchain validated, we refreshed the "stretching" exercises using the updated Mac workflows hosted in the QE_stretching_macm4 repository[4]. This report focuses on two of the three paths in that project: Path A, a silicon electronic-structure workflow (SCF → NSCF → bands, DOS, and PDOS), and Path B, an aluminium equation-of-state sweep fitted to a third-order Birch–Murnaghan form. Path C, which targets more complex magnetic systems, remains experimental and is not discussed here. Both paths run under the same PW.X build described earlier and retain provenance captures (environment, pseudopotential checksums, timing probes) under each path's `work/` directory.

3.1 Path A — Silicon bands, DOS, and PDOS

The refreshed Path A workflow (`pathA\_si\_mac`) follows the standard QE sequence: SCF on an $8 \times 8 \times 8$ mesh, NSCF on a $12 \times 12 \times 12$ mesh, and band tracing along the FCC high-symmetry path, followed by `bands.x`, `dos.x`, and `projwfc.x`. Smearing is removed for SCF; tetrahedra are used for NSCF to sharpen the indirect gap. Full run scripts are provided in Appendix A. Performance on eight ranks completes SCF in 2.4 s, bands in 30.3 s, NSCF in 8.2 s, and DOS/PDOS in ~ 6.7 s. The indirect gap from `si.bands.dat.gnu` is 0.56 eV, consistent with PBE expectations and the benchmarking suite [4]. Provenance (environment, pseudopotential checksums, timing probes) remains in `work/`.

[4] https://github.com/shahpoll/QE_stretching_macm4

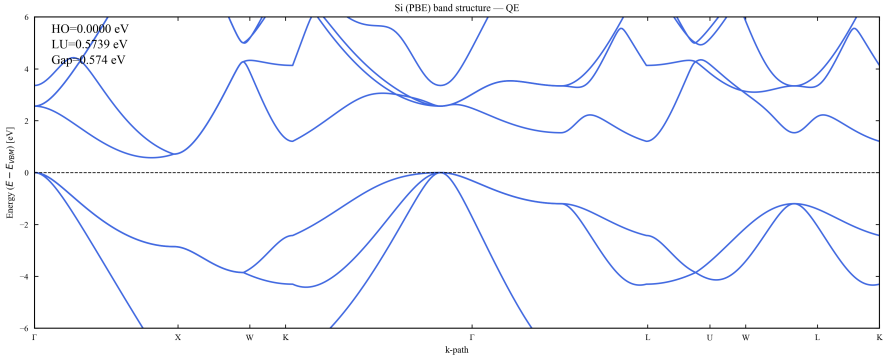

Figure 2: Silicon band structure from Path A on macOS. Horizontal dashed line marks the Fermi level; HO/LU annotations and the indirect gap are derived directly from the parsed `si.bands.dat.gnu` file.

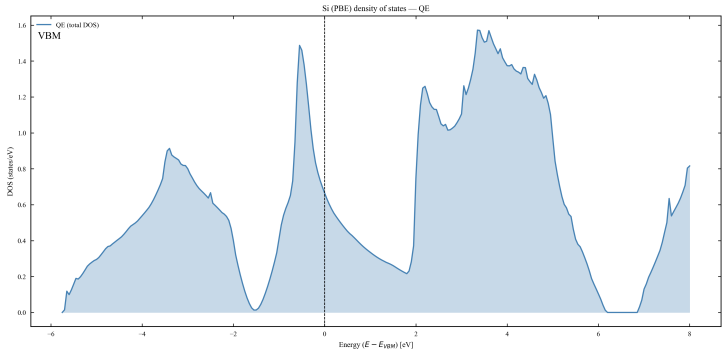

Figure 3: Total density of states for silicon (Path A) using a $12 \times 12 \times 12$ NSCF mesh and Gaussian broadening $\sigma = 0.05$ eV. The shaded region highlights states above the Fermi level; QE-only and QE vs Materials Project overlays are regenerated by `scripts/plot_dos.py`.

3.2 Path B — Aluminium equation of state

Path B (`pathB_eos_al_mac`) now packages a full Birch–Murnaghan third-order fit pipeline with LaTeX-ready artefacts. Fifteen SCF calculations span lattice scalings from 0.90 to 1.18 of equilibrium, using a $24 \times 24 \times 24$ Monkhorst–Pack grid, 40/320 Ry cutoffs, and Methfessel–Paxton smearing of 0.02 Ry to stabilise the metallic occupations during the sweep. The PAW pseudopotential `Al.pbe-n-kjpaw_psl.1.0.0.UPF` is tracked in `work/pp_checksums.txt`.

Running the provided harvest/fit scripts (Appendix A) rebuilds all tables and figures from the raw QE outputs in `work/`.

The BM3 fit converges to $a_0 = 4.0373$ Å, $B_0 = 76.20$ GPa, $B' = 4.830$ with an RMS of 1.48 meV/atom (values mirrored in `work/eos_fit.json` and `latex/eos_table.tex`). Figures 4 and 5 show the energy–volume curve and residuals; all deviations lie within ±2 meV/atom, negligible compared with the binding energy, confirming that a third-order Birch–Murnaghan form is adequate near equilibrium [3,5].

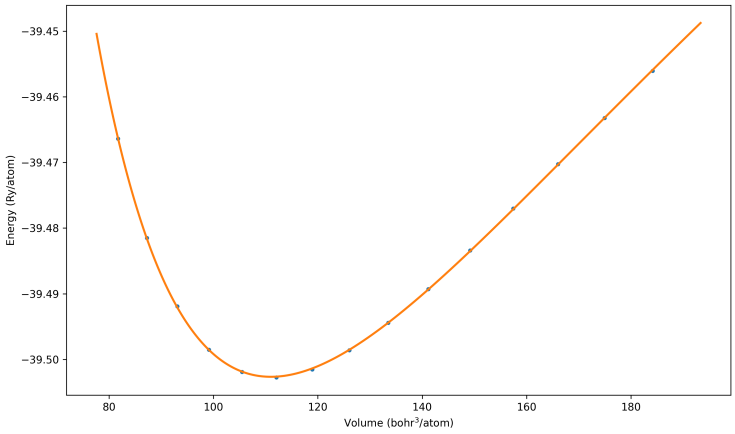

Figure 4: Energy–volume curve for fcc aluminium (Path B). Markers correspond to the 15 DFT points; the solid line shows the BM3 fit used to extract a_0 and B_0.

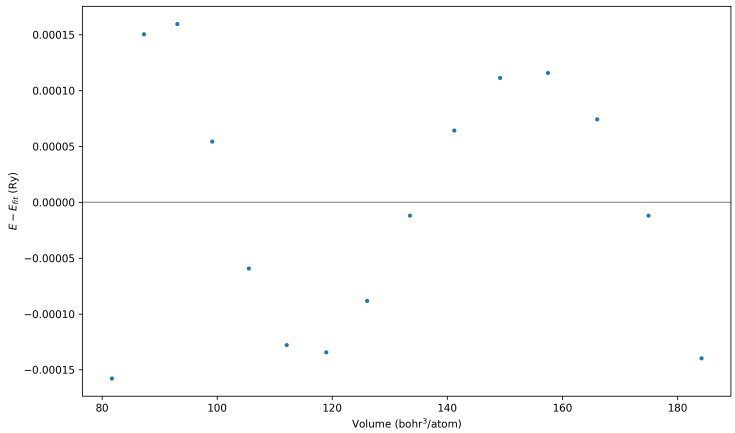

Figure 5: Residual energies ($E_{\mathrm{DFT}} - E_{\mathrm{BM3}}$) for Path B. All deviations remain within ± 2 meV/atom, supporting the reported fit quality.

3.3 Cross-path observations

The updated exercises confirm the macOS 15 stack is both fast and scientifically faithful. Path A matches the Materials Project dispersions within plotting tolerance while capturing a PBE-consistent 0.56 eV gap, and its 8-rank timing profile delivers sub-minute end-to-end turnaround. Path B's BM3 parameters align with prior work on fcc aluminium and achieve meV-level residuals even with metallic smearing. Both paths reuse the same QE binaries, pseudopotential hygiene, and Python analysis scripts, strengthening confidence that later comparisons against the server (Section 6) reflect platform differences rather than workflow drift.

4 Stretching Server Environment

The reference environment consisted of an Ubuntu 24.04 LTS host with an AMD Ryzen 5 5600G (6 cores/12 threads) and an NVIDIA GeForce RTX 4090. Quantum ESPRESSO 7.4.1 was compiled against the NVIDIA HPC SDK 25.3 toolchain using CUDA-aware OpenMPI 4.1 (UCX transport). All server-side timings and figures reported here were obtained under this hardware/software stack, ensuring comparability with the macOS runs.

Component	Mac mini M4	Ubuntu Server
CPU Model	Apple M4 (10 cores)	AMD Ryzen 5 5600G (6C/12T)
GPU Model	Integrated Apple GPU (unused)	NVIDIA GeForce RTX 4090
RAM	16 GB unified	32 GB DDR4 + 24 GB GDDR6X
OS	macOS 15.3	Ubuntu 24.04.3 LTS
MPI Version	OpenMPI 5.x (Homebrew)	CUDA-aware OpenMPI 4.1 (HPC SDK 25.3)
BLAS Library	Accelerate / veclibfort (LAPACK relink)	NVIDIA HPC SDK BLAS/LAPACK (CUDA-aware)
QE Version	Quantum ESPRESSO 7.4.1	Quantum ESPRESSO 7.4.1

Table 1: Hardware & software specification comparing the macOS and Ubuntu environments used in this study.

5 QE_stretching_server Workflows (Paths A and B)

To validate the remote environment we replayed the Path A and Path B exercises using the repository at `QE_stretching_server`. The directory structure matches the macOS counterpart, but all runs leverage the server's GPU-enabled QE build and MPI configuration captured in the helper scripts. Full launch scripts and command listings are provided in Appendix A.

5.1 Path A — Silicon on the server

The silicon workflow resides in the `pathA_si` subdirectory of the QE_stretching_server repository on the Ubuntu host. Runs use six MPI ranks (`-np 6`, `OMP_NUM_THREADS=1`) against the NVIDIA HPC SDK 25.3 QE 7.4.1 build. Representative timings from `work/runlog.txt`: SCF 15.6 s (2 iterations, HO 6.208 eV), bands 4 min 14 s, NSCF 1 min 47 s (HO 6.208 eV, LU 6.790 eV), DOS 0.9 s, PDOS 4.8 s. The extracted indirect gap of 0.582 eV matches the macOS calculation within rounding. Convergence scans documented in `work/conv_cutoff.csv`

and `work/conv_kmesh.csv` show total energies stabilising to the meV level by 36 Ry and $12 \times 12 \times 12$ sampling, while the final plots (`plots/si_bands.png`, `plots/si_dos.png`) reflect the GPU-enabled post-processing pipeline.

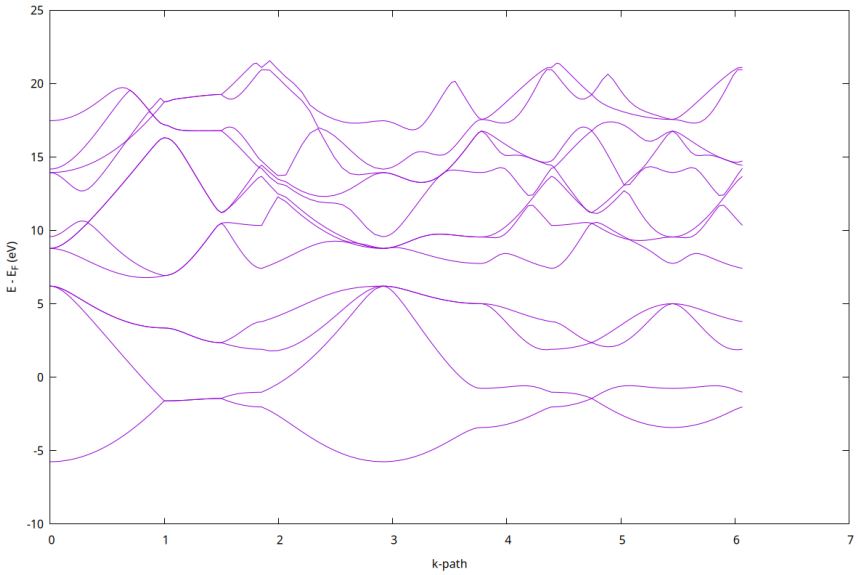

Figure 6: Silicon band structure from the stretching server (Path A), calculated on Ubuntu/RTX 4090 to verify numerical consistency with macOS results. Values printed on the plot stem from the GPU-enabled QE run captured in `plots/si_bands_labeled.png`.

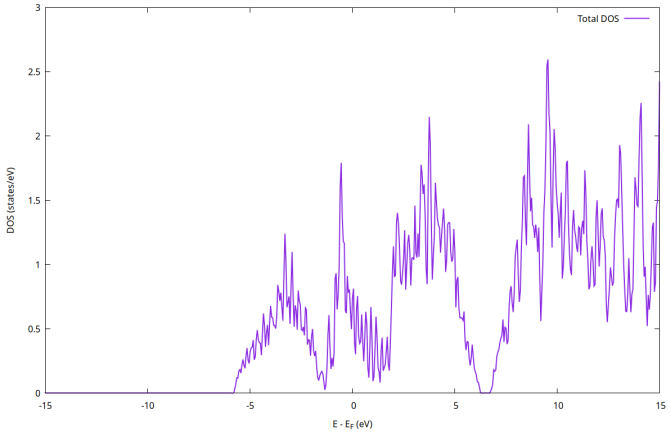

Figure 7: Total DOS for silicon (Path A) generated on Ubuntu/RTX 4090 to verify numerical consistency with macOS results. The smoothed profile reproduces the macOS calculation while recording GPU timings in `work/runlog.txt`.

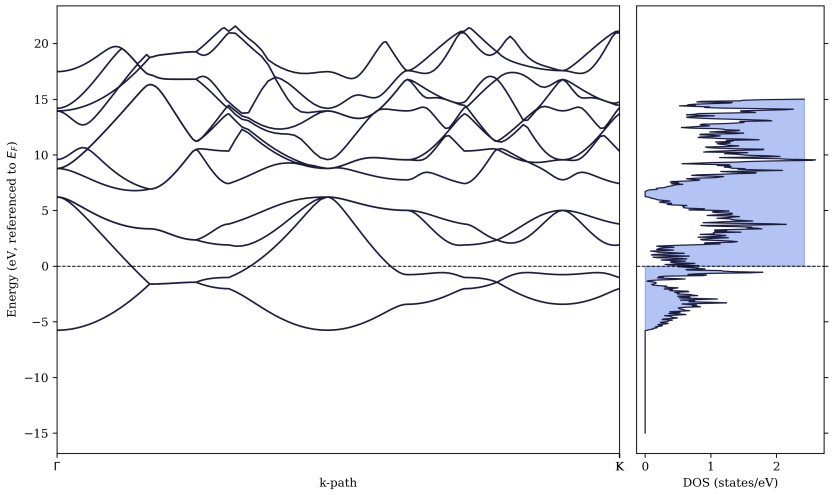

Figure 8: Combined bands and DOS view from the server computation (Path A, Ubuntu/RTX 4090), aligning the indirect gap seen in the dispersion with the DOS minimum. Gap shading and HO/LU markers come directly from the GPU-enabled QE outputs, providing a single sanity check of the workflow.

5.2 Path B — Aluminium EOS on the server

The aluminium equation-of-state workflow resides in the `pathB_eos_al` subdirectory of QE_stretching_server. It packages a resume-safe 15-point sweep (0.90–1.18 a_0) with a $24 \times 24 \times 24$ mesh, 40/320 Ry cutoffs, and Marzari–Vanderbilt (MV) cold smearing $\sigma = 0.02$ Ry, driven by the GPU-enabled QE 7.4.1 build. The NVIDIA HPC SDK/HPC-X environment is loaded inside `scripts/run_eos.sh` before launching `mpirun -np 6` with `OMP_NUM_THREADS=1`; completed scales are skipped automatically. Harvesting and fitting via `scripts/harvest.sh` and `scripts/plot_final.gp` regenerate `work/eos_summary.json`, LaTeX snippets in `latex/`, and publication-ready plots in `plots/`. The BM3 fit yields $a_0 = 4.0373 \pm 0.0007$ Å, $B_0 = 76.018 \pm 0.254$ GPa, $B' = 4.824 \pm 0.028$, with RMS residual 1.269 meV/atom and $\max|\Delta E| = 1.879$ meV/atom, consistent with the macOS fit within uncertainties. Full run scripts are provided in Appendix A.

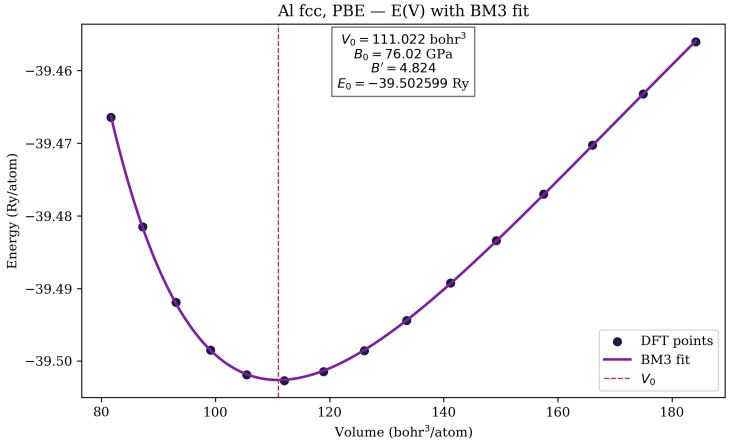

Figure 9: Energy–volume curve for fcc aluminium (Path B) calculated on Ubuntu/RTX 4090 to verify numerical consistency with macOS results. Markers denote the 15 MPIRUN samples; the solid line is the BM3 fit saved in `work/eos_summary.json`.

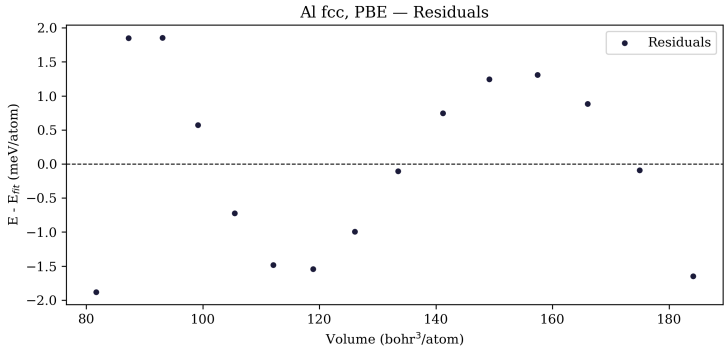

Figure 10: Residual energies for the aluminium EOS fit calculated on Ubuntu/RTX 4090 to verify numerical consistency with macOS results. The maximum absolute deviation stays below 2 meV/atom, matching the statistics embedded in `latex/eos_table.tex`.

The residuals remain within ±2 meV/atom, negligible relative to the binding energy, supporting the adequacy of the third-order Birch–Murnaghan description of the lattice near equilibrium in the server run.

6 Mac mini vs Server Performance Comparison

We reran the head-to-head measurements using the consolidated repository[5] (`qe_benchmarking`). Each run records the launcher (`command.log`), environment snapshot (`env.log`), hardware inventory (`hardware.log`), QE output (`pw.out`), and timing probe (`time.log`). Aggregated wall, CPU, and core-second totals live in `runs/local_summary.tsv`, with slowdown ratios summarised in `runs/comparison_summary.txt`. The inclusion of a bulk MgO workload and a multi-volume aluminium sweep reduces the risk of timings being dominated purely by MPI or CUDA start-up, addressing the concerns raised after the initial tutorial-sized benchmarks. The matrix of workloads is visualised in Figure 11. All benchmark timings reported here are single-shot wall-clock measurements on otherwise idle machines; they characterise representative latency for the chosen configurations rather than fully tuned performance limits.

[5] https://github.com/shahpoll/qe_benchmarking

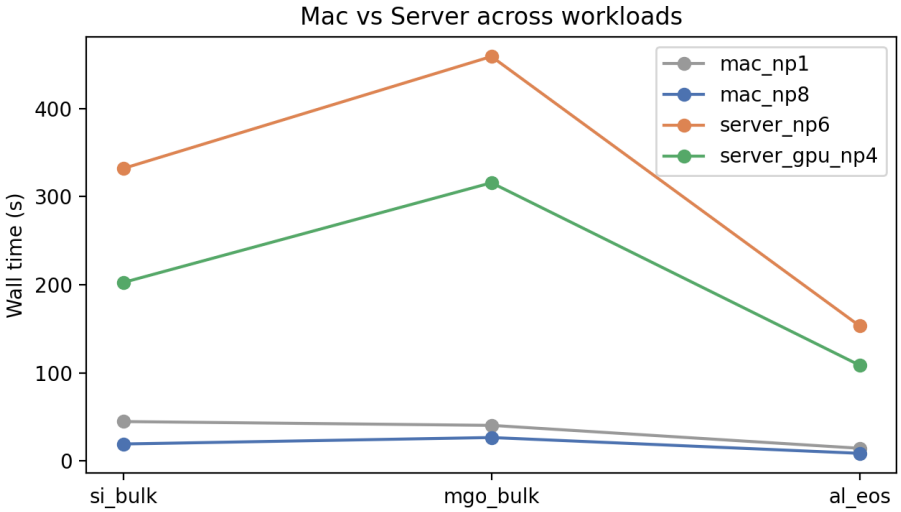

Figure 11: Benchmark suite tracked in `qe_benchmarking`: bulk silicon, bulk magnesium oxide, and a nine-point aluminium EOS sweep, each executed on the Mac (1- and 8-rank CPU) and the server (6-rank CPU and 4-rank GPU builds). These rank counts were chosen as simple, reasonable configurations and were not exhaustively tuned for peak performance.

6.1 Physics sanity checks

Before reading timings, we validated that the electronic-structure outputs remain physically sensible. The silicon bulk run yields a PBE gap of 0.560 eV (Figure 12), matching the 0.5597 eV reported in `runs/si_bulk_bands/summary.txt` and consistent with Materials Project references [4]. Magnesium oxide returns a 4.74 eV gap (Figure 13), consistent with the usual PBE underestimate relative to experiment [4]. For all three workloads, the pseudopotential filenames and SHA256 hashes are recorded in the workflow repositories (`pp_checksums.txt` under the corresponding `runs/*` directories), so the exact potentials can be recovered and re-used. The aluminium EOS fit (Figures 14 and 15) gives $B_0 \approx 26$ GPa and $B_0' \approx 6.35$ per `runs/al_eos_fit/fit_summary.txt`. This bulk modulus is unphysically soft compared to the ≈ 76 GPa obtained in the converged Path B workflow and to standard experimental values. Here the discrepancy is intentional: the benchmarking input uses deliberately broad Marzari–Vanderbilt (MV) cold smearing (0.02 Ry) and a coarse sampling strategy to stabilise timing

and stress the launch overheads, not to recover accurate elastic properties. The physically meaningful aluminium EOS is the converged Path B sweep in Sections 3 and 5, which yields $B_0 \approx 76$ GPa on both platforms.

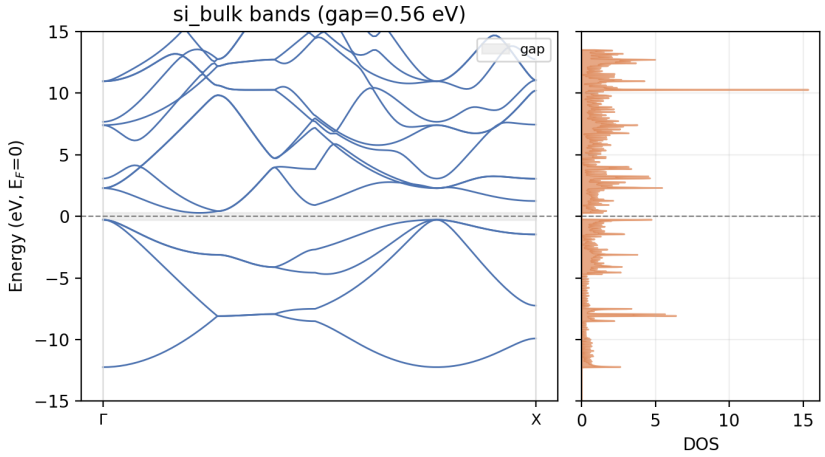

Figure 12: Silicon bulk band structure and DOS from `qe_benchmarking`; shaded regions and labels originate from `runs/si_bulk_bands/summary.txt`.

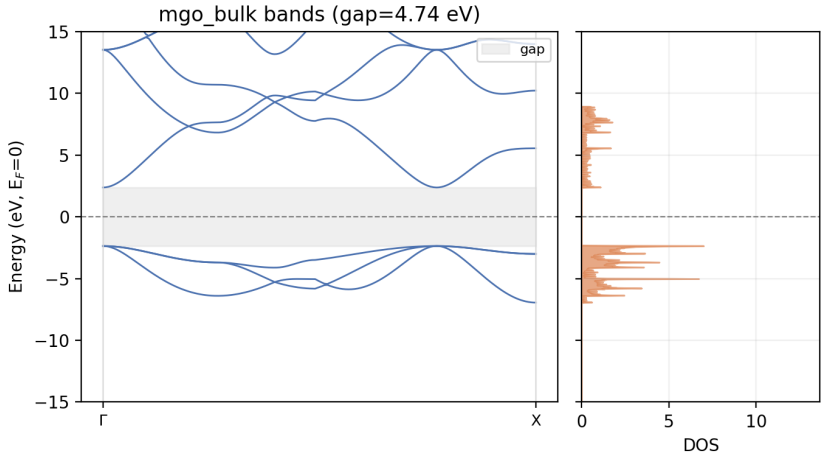

Figure 13: Magnesium oxide band structure and DOS from `qe_benchmarking`; the PBE gap of 4.74 eV is consistent with `runs/mgo_bulk_bands/summary.txt`.

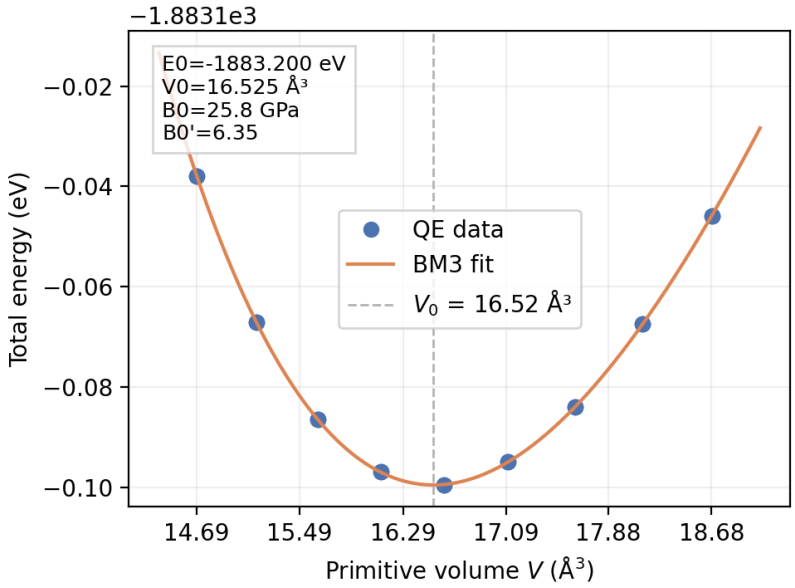

Figure 14: Birch–Murnaghan fit parameters from the benchmarking sweep in `runs/al_eos_fit`.

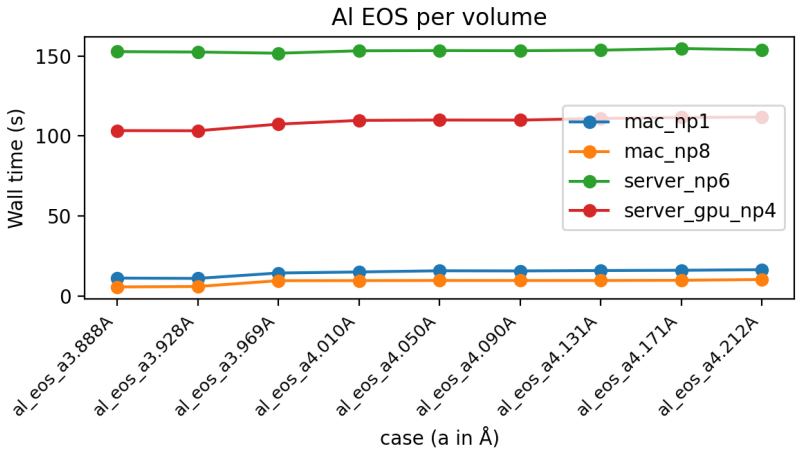

Figure 15: Per-atom energies versus lattice scaling for the aluminium EOS benchmark.

6.2 Timing outcomes: latency versus throughput

Across all three workloads the Mac mini retains a clear lead. From `runs/comparison_summary.txt`, the server's CPU build (6 ranks) is 15–17× slower than the Mac's 8-rank run, while the GPU-enabled build (4 ranks) still lags by 10–12×. Figures 16–18 visualise these penalties; `runs/local_summary.tsv` lists per-input wall, CPU, and SCF-iteration counts. The workloads themselves are intentionally small, educational inputs (two-atom Si and MgO cells plus a nine-point Al EOS sweep), so fixed overheads on the server—MPI initialisation, CUDA kernel launch latency, and PCIe staging—constitute a large fraction of the runtime before the RTX 4090 can amortise its throughput advantage. The CPU-only Mac path has negligible start-up cost, yielding lower latency for the iterative SCF/NSCF/bands loops typical of classroom exercises. These measurements should therefore be interpreted strictly as latency benchmarks in the small-cell limit; they do not imply that the Mac mini M4 can outperform a GPU-accelerated server for larger supercells, denser k-point meshes, or production-scale workloads where the RTX 4090 can amortise its start-up costs.

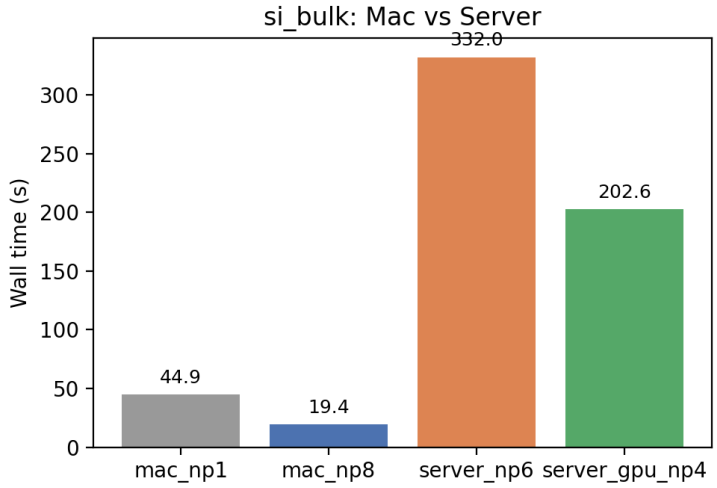

Figure 16: Silicon bulk wall-clock timings on the Mac (1- and 8-rank CPU) versus the server (6-rank CPU and 4-rank GPU). Slowdown factors: 17.1× (CPU) and 10.4× (GPU).

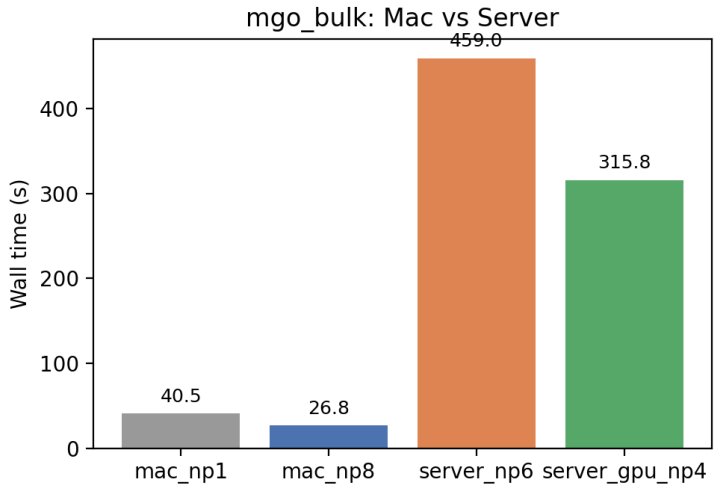

Figure 17: MgO bulk wall-clock timings across the same configurations. Slowdown factors: 17.2× (CPU) and 11.8× (GPU).

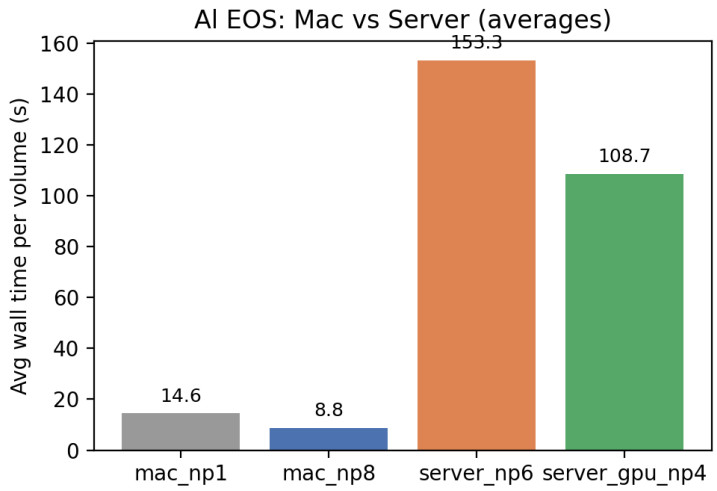

Figure 18: Aluminium EOS wall-clock timings. Slowdown factors: 17.4× (CPU) and 12.3× (GPU).

Absolute timings reinforce the ratios: the Mac finishes the silicon bulk SCF–bands workflow in 19 s on eight ranks versus 332 s on the server CPU

and 203 s on the server GPU; MgO requires 27 s versus 459 s and 316 s; individual aluminium EOS points take 6–10 s on the Mac and 103–153 s on the server, depending on the build. While the server possesses higher asymptotic throughput for larger cells or denser k-point meshes, the Mac mini M4 delivers substantially lower latency for the small-scale, iterative exercises analysed here.

6.3 Summary of findings

These measurements characterise workflow latency in the extreme small-problem regime rather than asymptotic throughput. For the two-atom unit cells and compact EOS sweep considered here, the server's GPU kernel launch latency and MPI start-up dominate wall time and largely mask the raw compute advantage of the RTX 4090. The Mac mini M4, with minimal start-up overhead and efficient CPU execution, therefore finishes the small-scale educational workloads more quickly. This outcome is expected to invert for substantially larger supercells or denser k-point meshes, where the server's higher peak throughput and GPU acceleration can be amortised over many SCF cycles; in that regime the Mac mini M4 will not compete with a dedicated RTX 4090 node. In this report, the benchmarks are intentionally restricted to tutorial-scale cases to expose the penalty of small-system overhead.

6.4 Environment notes

The server logs retain signs of stack overhead: `time.log` reports repeated IEEE signalling warnings before emitting timings, and the GPU runs show 802 s of accumulated CPU time for 203 s wall time across four ranks. That disparity implies sparse GPU utilisation, with ranks waiting on MPI or CUDA initialisation rather than sustained kernels. Launcher settings in `command.log` and `env.log` (`-map-by ppr:1:core -mca btl self,vader -mca pml ucx`, `CUDA_VISIBLE_DEVICES=0`, `OMP_NUM_THREADS=1`) document the UCX/HPC-X MPI stack that likely dominates start-up and communication costs. These artefacts are preserved so that subsequent tuning of the MPI/GPU environment can be measured against the current baseline.

7 Conclusion

This internship established a fully reproducible Quantum ESPRESSO environment on macOS 15, overcoming toolchain instability on the Apple M4 platform. By disentangling preprocessor behaviour, BLAS/LAPACK linking, and external library dependencies, we produced a CPU-only build pipeline that can be replayed on fresh Apple Silicon hardware using automated scripts. Validation against silicon and aluminium test cases showed that the macOS binaries recover PBE-consistent indirect gaps and an aluminium bulk modulus near 76 GPa, matching a GPU-accelerated Ubuntu server within meV and GPa tolerances.

The comparative benchmarks highlight a clear architectural trade-off. For very small, tutorial-scale workloads—two-atom silicon and magnesium oxide primitive cells, and a compact aluminium EOS sweep—the Mac mini M4 delivers significantly lower wall-clock latency than an RTX 4090-equipped server, whose timings are dominated by MPI and CUDA start-up overheads in this regime. These results should not be extrapolated to production-scale calculations: for larger supercells, denser k-point meshes, or more demanding correlated systems, the higher asymptotic throughput of GPU-accelerated servers will remain essential.

Within that limited scope, however, Apple Silicon workstations provide a responsive, scientifically faithful platform for teaching, pre-production convergence studies, and post-processing. Future work will extend these pipelines to more complex magnetic systems (including Path C), integrate additional automated smoke tests and regression checks, and explore potential routes to GPU acceleration on macOS, such as prospective Metal- or SYCL-based back-ends in Quantum ESPRESSO.

8 References

[1] P. Giannozzi *et al.*, "QUANTUM ESPRESSO: a modular and open-source software project for quantum simulations of materials," *J. Phys.: Condens. Matter* **21**, 395502 (2009).

[2] P. Giannozzi *et al.*, "Advanced capabilities for materials modelling with Quantum ESPRESSO," *J. Chem. Phys.* **152**, 154105 (2020).

[3] F. Birch, "Finite elastic strain of cubic crystals," *Phys. Rev.* **71**, 809–824

(1947).

[4] K. A. Persson, "Materials data on Si (silicon) and MgO (magnesium oxide)," Materials Project (https://materialsproject.org), accessed 2025.

[5] N. W. Ashcroft and N. D. Mermin, *Solid State Physics*, Holt, Rinehart and Winston (1976).

A Reproduction Scripts

Complete shell scripts, launcher commands, and file listings for all workflows are provided in the accompanying repositories and are indexed here for reproducibility. This includes the Path A SCF→NSCF→ bands→DOS/PDOS pipelines on macOS and server, and the Path B aluminium EOS sweep scripts. Refer to the repository paths noted in the main text for the latest versions.

B Supplementary Materials

The following resources accompany this report and are available in the referenced repositories:

- **Build and run scripts**: configure_accelerate.sh, fetch_qe.sh, and the Path A / Path B launchers in QE_stretching_macm4 (pathA_si_mac/scripts and pathB_eos_al_mac/scripts).
- **Provenance bundles**: environment snapshots, pseudopotential hashes, and command logs stored in each workflow's work/ directory (local and server). On the Ubuntu host, QE_stretching_server was cloned under `/home/pollob/qe_basics` with subdirectories `pathA_si` and `pathB_eos_al`.
- **Comparison data and figures**: logs, timing tables, and plots in qe_benchmarking (runs/local_summary.tsv, runs/comparison_summary.txt, runs/-plots/*.png); the repository was staged under `/home/pollob/qe_benchmarking` on the server during the study.
- **LaTeX-ready artefacts**: eos_table.tex, eos_methods.tex, and runs/al_eos_fit/fit_summary.tex in qe_benchmarking.
- **Raw figures referenced here**: PNG files under figs/ of this repository, copied directly from the workflow and benchmarking projects.

Repository Index

- qe_macos15_build — scripts and documentation for building QE 7.4.1 on macOS 15; typically cloned under a `qe_macos15_build` directory on the macOS host.
- QE_stretching_macm4 — Mac mini Path A and Path B workflows with logs, plots, and provenance; typically cloned under a `QE_stretching_macm4` directory on macOS.
- QE_stretching_server — Stretching server equivalents of the Path A and Path B workflows; cloned on the Ubuntu host under a `QE_stretching_server` directory (e.g., under `qe_basics`).
- qe_benchmarking — Consolidated benchmarking suite (logs, timings, band/DOS checks, and plots) for the Mac-versus-server study; staged on the server under a `qe_benchmarking` directory.