

TorchRef: an open-source *PyTorch* framework for crystallographic refinement

Hans-Peter Seidel,* Jörg Standfuss* and Tobias Weinert*

Center for Life Sciences, Paul Scherrer Institut, 5232 Villigen PSI, Switzerland. *Correspondence e-mail: hans.seidel@psi.ch, joerg.standfuss@psi.ch, tobias.weinert@psi.ch

Received 22 April 2026

Accepted 3 September 2026

Edited by T. Lane, Deutsches Elektronen
Synchrotron, Germany

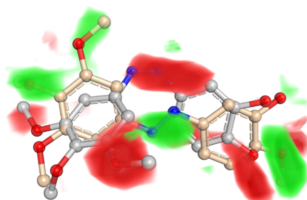
Keywords: crystallographic refinement;
PyTorch; *TorchRef*; automatic differentiation;
time-resolved crystallography; GPU
acceleration.

Supporting information: this article has
supporting information at journals.iucr.org/j

Macromolecular crystallographic refinement is central to structural biology, with a long history of methodological development. Differentiable-programming frameworks now offer a natural base for extending this toolbox with new targets and parameterizations, with access to hardware acceleration. Here we present *TorchRef*, a *PyTorch*-based refinement framework that exposes all refinable parameters (atomic coordinates, atomic displacement parameters, occupancies and scale factors) to automatic differentiation. The framework implements fast Fourier transform based structure factor calculations, the French–Wilson treatment of intensities, bulk-solvent modelling and stereochemical restraints from the CCP4 Monomer Library. A modular target architecture allows loss functions to be combined, weighted and extended independently of the core refinement machinery. Across 725 *Phaser*-placed *AlphaFold* predictions refined head to head with *phenix.refine*, *REFMAC5* and *TorchRef*, the three engines reach median R_{free} values within 0.30 percentage points of each other while maintaining comparable geometry. *TorchRef*'s structure factor calculation is comparable to existing methods and benefits from multiple CPU cores and GPU acceleration. *TorchRef*'s differentiable, modular design also enables workflows beyond conventional refinement: as an example, we implemented direct refinement against amplitude differences for time-resolved data, where the refined model reproduced the observed difference density more closely than the deposited extrapolation-refined model. *TorchRef* is released under the MIT licence with full API documentation and tutorials, providing an accessible platform for developing and testing new crystallographic refinement protocols.

1. Introduction

Macromolecular crystallographic refinement is a mature and indispensable component of structural biology and was central to building the models that inform our understanding of protein structures hosted in the Protein Data Bank (PDB) (Burley *et al.*, 2019). Established refinement software combines well optimized parameterization, chemically informed restraint libraries and cross-validated maximum-likelihood targets to balance prior knowledge against the data (Pannu & Read, 1996; Turk, 2013). This stabilizes refinement in challenging regimes such as low resolution, model incompleteness and complex macromolecular assemblies, and is increasingly complemented by physically motivated force fields (Moriarty *et al.*, 2020). The major suites in current use include *Phenix* (Adams *et al.*, 2010; Liebschner *et al.*, 2019), *REFMAC5* (Murshudov *et al.*, 2011; Murshudov *et al.*, 1997) and *BUSTER* (Blanc *et al.*, 2004), which remain under active development, as shown by the recent release of *Phenix 2.0* and the ongoing modernization and reimplementations of *REFMAC5* through *gemmi* and *Servalcat* (Yamashita *et al.*, 2023).



OPEN ACCESS

Published under a CC BY 4.0 licence

At the same time, experimental modalities have diversified. Microcrystal electron diffraction enables structure determination from nanometre-sized crystals (Nannenga & Gonen, 2019; Clabbers & Xu, 2021). X-ray free-electron lasers and upgraded synchrotrons have enabled serial crystallography, which collects tens of thousands of still images from micro-metre-sized crystals and merges them into complete datasets while mitigating radiation damage, making room-temperature measurements routinely accessible (Schlichting, 2015; Chapman, 2019; Barends *et al.*, 2022). Time-resolved serial crystallography extends these methods to the recovery of structural intermediates spanning femtosecond-to-second timescales and has revealed reaction pathways in diverse light-triggered systems (Brändén & Neutze, 2021; Schulz *et al.*, 2022; Hekstra, 2023; Khusainov *et al.*, 2024). These modern crystallographic experiments often produce large, heterogeneous datasets that call for refinement frameworks capable of handling ensembles, mixed occupancies and custom priors (Henning Hansen *et al.*, 2026).

Modern experiments increasingly aim to elucidate macromolecular function rather than to determine structure, a modality for which the established, general-purpose refinement software was not designed. A growing ecosystem of specialized tools has emerged to bridge this gap. Some integrate molecular dynamics into refinement workflows (Moriarty *et al.*, 2020), while others use crystallographic data to restrain molecular-dynamics simulations (Mikhailovskii *et al.*, 2024) or couple quantum-mechanical calculations to refinement (Lundgren *et al.*, 2024). A separate line of work targets difference electron density: improving the maps themselves (Fadini *et al.*, 2025), using them for state identification (Wickstrand *et al.*, 2020) and supporting refinement of the weakly populated, transient states they reveal in time-resolved crystallography (De Zitter *et al.*, 2022).

In parallel, the broader machine-learning community has converged on differentiable-programming frameworks, most prominently *PyTorch* and *JAX*, that fundamentally change how numerical algorithms are developed and optimized (Paszke *et al.*, 2019; Bradbury *et al.*, 2018). In these frameworks, all numerical operations are performed on multi-dimensional arrays (tensors) and recorded in a dynamic computational graph. When a scalar objective, such as a likelihood function, is evaluated, the framework traverses the graph in reverse to compute exact gradients with respect to every parameter, a process known as automatic differentiation or backpropagation. For crystallographic refinement, this means that new target functions, parameterizations or restraint terms can be implemented without ever deriving or coding analytical gradients by hand. Furthermore, because the same tensor operations execute on both CPUs and GPUs without code changes, algorithms can benefit from hardware acceleration depending on their computational structure.

Within structural biology, usage of these frameworks is gaining traction (AlQuraishi & Sorger, 2021; Gasparotto *et al.*, 2024). Recent work couples machine-learning-accelerated quantum-mechanical potentials with crystallographic and cryo-electron microscopy refinement, making the case for end-

to-end differentiable models in academic research (Zubatyyuk *et al.*, 2025). In a separate line of work, *OpenFold* has been used as a structural prior to aid refinement (Fadini *et al.*, 2026), building on *SFCalculator* (Li *et al.*, 2025), which evaluates structure factors within *PyTorch* and exposes them to automatic differentiation. These efforts demonstrate the value of differentiable methods in refinement, yet they typically address individual components or build on established packages rather than providing a fully differentiable refinement framework.

Here, we introduce *TorchRef*, a *PyTorch*-based macromolecular refinement framework in which all refinable parameters (atomic coordinates, displacement parameters, occupancies and scale factors) are exposed to automatic differentiation. *TorchRef* is designed primarily as a platform for method development rather than as a replacement for established packages, while still achieving refinement quality comparable to that of widely used programs. It supports fast Fourier transform (FFT)-based structure factor calculation and benefits from the associated $O(N \log N)$ scaling. The framework includes French–Wilson treatment of intensities (French & Wilson, 1978), bulk-solvent and anisotropic scaling following established implementations (Adams *et al.*, 2010; Blanc *et al.*, 2004; Afonine *et al.*, 2005), and stereochemical restraints derived from the CCP4 Monomer Library and available via the GitHub repository at <https://github.com/MonomerLibrary/monomers>. Full API documentation of *TorchRef* and tutorials are available at <https://github.com/HatPdotS/TorchRef>. *TorchRef* is under active development; this paper describes v0.6.4.

2. Results

2.1. *TorchRef* module structure

TorchRef implements a crystallographic refinement pipeline as a collection of differentiable *PyTorch* modules [Fig. 1(a)]. This section sketches the conceptual role of each module and the design decisions that shape the pipeline; functional forms, default parameters and implementation details are collected in the *Supplementary methods* (see the supporting information), and the full API is documented with the source code.

The base module provides the shared input/output and numerical primitives. Operations are expressed in native *PyTorch*, providing identical code paths on the CPU and GPU; dedicated kernels (device-specific low-level routines) are introduced as alternative paths for performance-critical operations, *e.g.* structure factor calculation (*Supplementary methods, Structure factor calculation*).

Experimental data are handled by the *ReflectionData* class. It carries the reflections together with their outlier and cross-validation flags, performs French–Wilson conversion when intensities are supplied (French & Wilson, 1978), and exposes the data and respective subsets via a property-style accessor syntax. MTZ and CCP4-map I/O is delegated to *reciprocalspaceship* and *gemmi* (Greisman *et al.*, 2021; Wojdyr,

2022), and reflections are stored in the CCP4 asymmetric unit; for details on data handling see *Supplementary methods, Reflection data preparation*. A companion class, `DatasetCollection`, organizes multiple datasets under a shared scale for time-resolved and ensemble workflows (*Supplementary methods, Multi-dataset and multi-model infrastructure*).

The `Model` object encapsulates all parameters relating to the crystallographic model: coordinates, atomic displacement parameters (ADPs) and occupancies, with support for selective freezing, group selection and reparameterization. Coordinates

are stored and refined in Cartesian coordinates (Å), with fractional coordinates generated on demand for structure factor evaluation; ADPs are stored in log-space (isotropic) or Cholesky-decomposed form (anisotropic) to enforce positivity and positive-definiteness without bound constraints; occupancies are parameterized at the residue level with optional sum-to-one constraints for alternate conformers (*Supplementary methods, Parameterization*).

The `Scaler` module estimates the bulk-solvent and overall scale contributions and combines them with the calculated structure factors before they reach the data target. Bulk

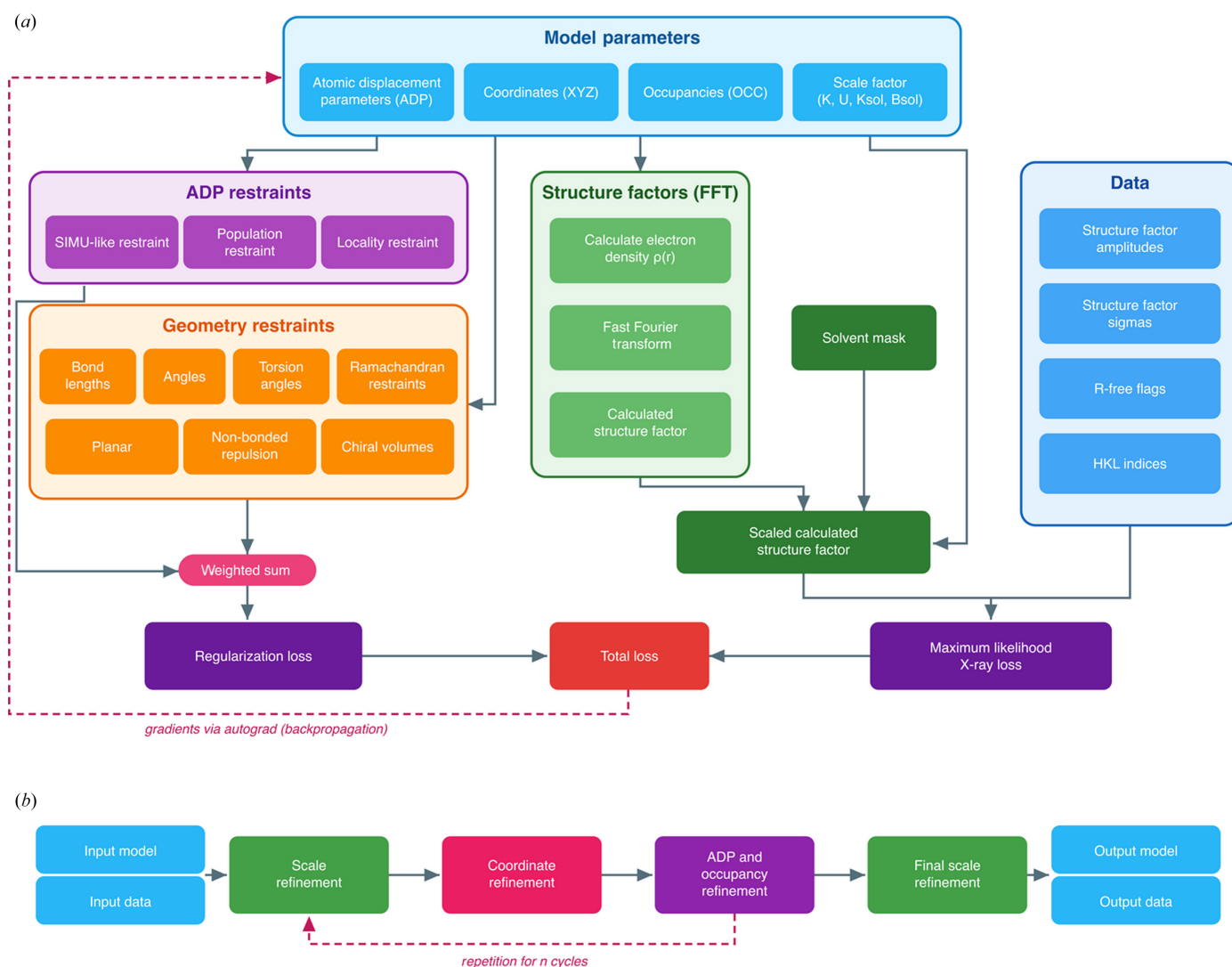


Figure 1

(a) Schematic overview of information flow during a forward pass in the standard *TorchRef* refinement loop. The central goal of any macromolecular refinement is to improve the data-model fit. Atomic coordinates are used to evaluate geometry-based restraint targets: bond lengths, bond angles, torsion angles, planarity, non-bonded repulsion, chiral volumes and an opt-in Ramachandran score (off by default and unused in the *AlphaFold* benchmark). ADPs are used to feed three regularization targets: a bonded similarity restraint, a shifted inverse-gamma population restraint on the *B*-value distribution and a kNN locality restraint on log *B*. From all model parameters, structure factors F_{calc} are computed, scaled against experimental data and compared with observed amplitudes F_{obs} through a maximum-likelihood target. Individual target values are aggregated according to their respective weights into a scalar loss, from which gradients are backpropagated to all refinable parameters via *PyTorch*'s autograd. (b) Overview of the *TorchRef* refinement workflow. The process begins by loading structure coordinates (PDB or mmCIF format) and reflection data (MTZ or CIF format). An initial scale refinement is performed, followed by the main refinement loop: scale update, coordinate refinement, and joint optimization of occupancies and ADPs. After the specified number of cycles, a final round of scale refinement is performed, followed by writing out the refined model and structure factors.

solvent is modelled by an exponentially attenuated flat mask following the approach of Afonine *et al.* (2005); the overall scale carries an anisotropic component together with a smooth resolution-dependent isotropic rescaling. The scaler is fitted with *PyTorch*'s L-BFGS optimizer using a separate α -free scale objective (default 1s, with `nll` and `ml_noalpha` available) rather than the α -centred body target, with a soft penalty on $\mathbf{U}_{\text{scaler}}$ that keeps the anisotropic correction minimal so it does not absorb *B*-value information that belongs in the atomic model (*Supplementary methods, Bulk-solvent mask and Scaling*).

Stereochemical restraints are built by the *Restraint* module from the CCP4 Monomer Library (Long *et al.*, 2017), with user-supplied CIFs overlaid where required. The restraint topology is constructed lazily on first access, after which atom indices are cached and per-restraint deviations are evaluated in a single vectorized pass. The full set of restraint targets (bonds, angles, planes, chirals, torsions, non-bonded repulsion with riding hydrogens and an opt-in Ramachandran target), together with their functional forms, is described in *Supplementary methods, Geometric restraints and Hydrogen handling for non-bonded repulsion*. A fully differentiable AMBER14/GAFF2 force-field target is provided as an experimental alternative; its construction and the bridge exposing OpenMM energies to *PyTorch*'s automatic differentiation (autograd) are described in *Supplementary methods, AMBER14/GAFF2 force-field target*.

Data targets and regularizers are implemented as dedicated modules that share a common `forward()` contract returning a scalar loss. The default X-ray target is the Read maximum-likelihood `ml` target (Read, 1986): a Rice/folded-normal likelihood centred on the Luzzati-coupled mean $\alpha|F_{\text{calc}}|$ ($D|F_{\text{calc}}|$ for unnormalized amplitudes). The coupling $\alpha = \sigma_A \sqrt{\Sigma_N / \Sigma_P}$ [where, in each resolution shell, $\Sigma_N = \langle F_o^2 / \epsilon \rangle - \langle \sigma_{F^2}^2 / \epsilon \rangle$ is the noise-corrected observed power and $\Sigma_P = \langle F_c^2 / \epsilon \rangle$ the model power, with ϵ the reflection multiplicity factor (*Supplementary methods, X-ray target functions*)] and total variance β (including the model-error component β_{model}) are derived from a per-shell maximum-likelihood estimate of σ_A on free reflections, using noise-corrected shell moments; the scale fit itself is restricted to α -free objectives because α is degenerate with the scale. Free reflections do not contribute direct coordinate or ADP gradients, but they do enter the detached shell error model. Consequently R_{free} remains a useful profile diagnostic but is not a strictly untouched held-out score. Several additional targets remain available for diagnostic, benchmarking and prototyping purposes (*Supplementary methods, X-ray target functions*). ADP regularization comprises a bonded similarity term, a *k*-nearest-neighbour (kNN) locality term on $\log B$ and a population-level shifted inverse-gamma restraint on the *B*-value distribution (*Supplementary methods, ADP restraints*).

A central *LossState* module tracks all active targets, applies the fixed per-group weights described in Section 2.2, enables parameter gradients on demand and integrates with the caching layer so that shared intermediates (structure factors, the solvent mask, restraint indices) are reused across

targets within a refinement step (*Supplementary methods, Caching infrastructure*).

The *Refinement* module [Fig. 1(b)] orchestrates macrocycles, tracks refinement statistics and writes outputs. We use L-BFGS as the default optimizer, mirroring *phenix.refine* (Afonine *et al.*, 2012); for future developmental projects *TorchRef* retains full compatibility with any other *PyTorch* optimizer. Macrocyclic scheduling, outlier rejection and the multi-resolution rigid-body protocol are described in *Supplementary methods, Optimization and refinement cycle*.

2.2. Component weighting

Balancing the X-ray likelihood with geometric restraints and ADP regularization is a long-standing challenge in crystallographic refinement. The default production components are formulated as negative log-likelihoods or prior penalties on the same loss scale (with least-squares modes retained as diagnostic alternatives). From first principles these terms can be summed; in practice, the residual mismatch between the data-fit, geometry and ADP-prior scales requires per-group weighting. *TorchRef* uses fixed per-group weights $w_{\text{xray}} = 1.0$, $w_{\text{geometry}} = 0.2$, $w_{\text{adp}} = 0.02$ (DEFAULT_GROUP_WEIGHTS) as the default; these were calibrated by a two-dimensional sweep on a subset of the *AlphaFold* benchmark (Section 2.3, Fig. S1). Component-level overrides are exposed at the command-line interface for users who wish to fine-tune the weights on a per-structure basis. Adaptive weighting schemes are planned for future versions of *TorchRef* (see the *Discussion*).

2.3. Validation of the refinement framework

To verify the functionality of *TorchRef*, we benchmarked it against *phenix.refine* and *REFMAC5* under matched protocols. To create a challenging and realistic test set, we used *AlphaFold* (Jumper *et al.*, 2021) predictions placed by molecular replacement and refined them against their respective experimental datasets.

Starting from a 1000-structure non-redundant set assembled from the PDB, we mapped each protein chain to its UniProt accession via the Research Collaboratory for Structural Bioinformatics (RCSB) Data API, downloaded the corresponding *AlphaFold*-DB model and trimmed it to the crystallized construct. Residues at which the *AlphaFold* sequence disagreed with the construct were truncated to alanine, except at proline and glycine positions, which were kept as is. The trimmed search models were pLDDT $\rightarrow$ B-converted with `phenix.process_predicted_model` invoked with `maximum_rmsd=100` so that no further confidence-based residue removal was applied beyond the sequence-based trimming above; the default pLDDT-to-B mapping of `phenix.process_predicted_model` was used. The processed models were placed by *Phaser* in MR_AUTO mode using the crystallized composition (McCoy *et al.*, 2007). *Phaser* solved 767 of 825 (93%) *AlphaFold*-complete structures; the solved subset defines the benchmark set used.

Placed *AlphaFold* models were then refined with *TorchRef*, *phenix.refine* (Liebschner *et al.*, 2019) and *REFMAC5* (Murshudov *et al.*, 2011) under matched protocols. All refined models, and the unrefined placed models for reference, were independently re-scored by *phenix.model_vs_data* against identical structure factors and free flags to produce the R factors reported here, holding the scoring conditions constant across engines. Geometry was assessed by *REFMAC5* zero-cycle scoring against a common restraint library so root-mean-square Z -score (RMSZ) values are comparable across engines.

Of the 767 *Phaser*-solved starting models, *Phenix*-validated refinements were obtained for 750 with *TorchRef*, 734 with *phenix.refine* and 760 with *REFMAC5*. The dominant failure modes were a special-position pathology in some *Phaser*-placed *AlphaFold* starting models (which is unscorable for

every engine and additionally caused *phenix.refine* to abort) and out-of-memory (OOM) errors at the 8 GB job limit for *TorchRef* on the largest structures (nine structures, all of which pass at 16 GB). The conserved set on which all three engines refined and *Phenix*-validated successfully comprises 725 structures; this is the set used for the headline endpoint comparisons below.

R factors. The median *Phenix*-validated R_{free} values were 0.3165 for *TorchRef*, 0.3161 for *phenix.refine* and 0.3135 for *REFMAC5* [Fig. 2(a)], all substantially below the unrefined prediction's 0.4083. The three engines therefore agree to within 0.30 percentage points, a spread far smaller than the separation between the refined and unrefined states, so at this level the differences are of little practical consequence for refinement quality. Median R_{work} values are 0.2695 for *TorchRef*, 0.2719 for *REFMAC5* and 0.2744 for *phenix.refine*:

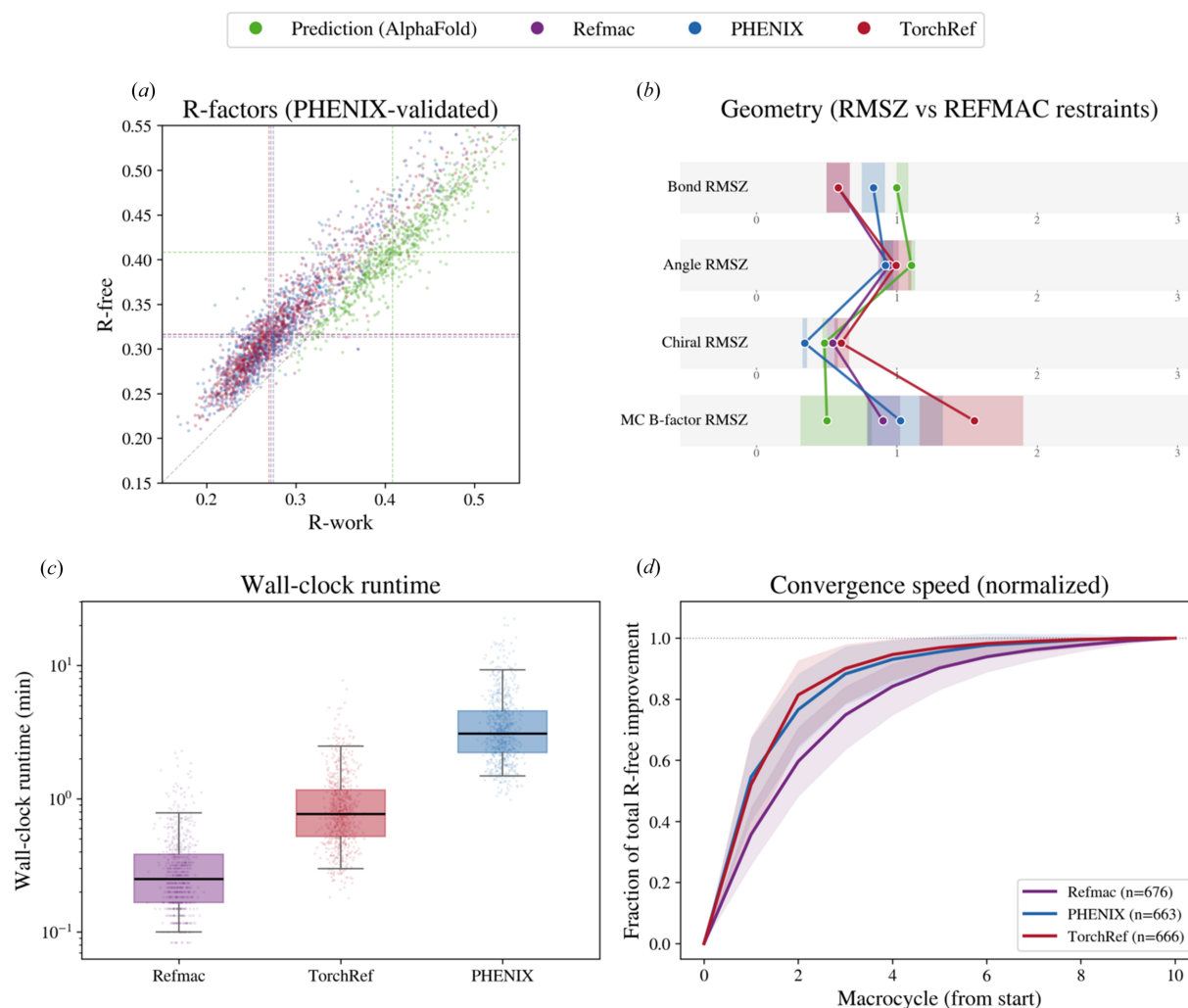


Figure 2

AlphaFold benchmark; 767 *Phaser*-solved candidates. Panels (a)–(c) use the conserved subset of 725 structures on which all three engines refined and *Phenix*-validated successfully. Panel (d) additionally excludes trajectories with less than 0.02 total R_{free} improvement, leaving 676 *REFMAC5*, 663 *phenix.refine* and 666 *TorchRef* trajectories. R factors are *Phenix*-validated under constant scoring conditions; geometry RMSZ is *REFMAC5*-validated against a uniform restraint library. Engines: unrefined *AlphaFold* prediction (green), *REFMAC5* (purple), *phenix.refine* (blue), *TorchRef* (red). (a) R_{free} versus R_{work} scatter. Dashed lines mark per-engine medians. (b) Geometry RMSZ strip (bond, angle, chiral, main-chain B value); point = median, band = 25–75th percentile. (c) Wall-clock runtime (log scale). (d) Convergence speed: fraction of total R_{free} improvement versus macrocycle, shown as the median across structures with a 25–75th percentile band.

TorchRef reaches the lowest R_{work} but the largest $R_{\text{work}}-R_{\text{free}}$ gap, 0.0470, above *REFMAC5* (0.0416) and *phenix.refine* (0.0417). This gap points to a mild tendency to overfit the working set. Broken down per structure, the paired median R_{free} difference is -0.0010 against *phenix.refine* and $+0.0030$ against *REFMAC5* (Fig. S2). The excess over *REFMAC5* is small but statistically robust ($p = 1.6 \times 10^{-14}$). *TorchRef*'s internal R_{free} scorer agrees with *phenix.model_vs_data* to within a few tenths of a percentage point, and a cross-scoring analysis on the *AlphaFold* benchmark (Fig. S3) confirms that no engine favours its own refinement output. We report *Phenix*-validated R factors from *phenix.model_vs_data* as a single external scorer applied uniformly to every engine output.

Geometry. Bond, angle, chiral and main-chain B -value RMSZ values were computed by *REFMAC5* against a uniform restraint library so the metric is comparable across engines [Fig. 2(b)]. *TorchRef*'s coordinate geometry is essentially on par with *REFMAC5*: bond, angle and chiral RMSZ are 0.58, 0.99 and 0.60 for *TorchRef*, and 0.58, 0.95 and 0.54 for *REFMAC5*. This agreement is consistent with both programs using the CCP4 Monomer Library restraint definitions. The remaining discrepancy is in the ADPs: *TorchRef*'s main-chain bonded- B RMSZ is 1.55, in contrast to 0.90 for *REFMAC5* and 1.03 for *phenix.refine*.

Weight calibration. The default per-group weights were chosen from a two-dimensional sweep of (w_{geometry} , w_{adp}) at fixed $w_{\text{xray}} = 1$ on a subset of the benchmark (Fig. S1). Bond/angle RMSZ tightens as w_{geometry} is increased (Fig. S1, top left), whereas the main-chain bonded- B RMSZ responds primarily to w_{adp} and is treated separately (Fig. S1, top right). Meanwhile, R_{free} shows a U-shaped dependence on w_{geometry} : it falls as the geometry weight is increased from very loose values, reaches a minimum at moderate geometry weights slightly below the chosen default and rises sharply once the restraint terms begin to dominate the X-ray likelihood (Fig. S1, bottom right). The $R_{\text{work}}-R_{\text{free}}$ gap closes as either weight is tightened (Fig. S1, bottom left), so the looser end of the sweep also carries the heaviest overfitting penalty. The chosen default (w_{geometry} , w_{adp}) = (0.2, 0.02) sits just to the tighter side of the R_{free} minimum, trading a small amount of R_{free} for tighter bond/angle geometry and a smaller $R_{\text{work}}-R_{\text{free}}$ gap. Pushing the restraint weights further past this point results in significantly worse R_{free} values, likely due to underfitting and convergence problems; adaptive weighting schemes are planned for a future release.

Convergence and runtime. The per-cycle program-reported R_{free} shows *TorchRef* and *phenix.refine* converging at essentially the same rate, both reaching $\sim 90\%$ of their total R_{free} improvement by macrocycle three and both faster in cycle count than *REFMAC5* [Fig. 2(d)]. Wall-clock per refinement [Fig. 2(c)] shows *TorchRef* 4.0 times faster than *phenix.refine* but 3.1 times slower than *REFMAC5*. In a single-core comparison (Fig. S4), *TorchRef* is 21.0 times slower than *REFMAC5* and 1.4 times slower than *phenix.refine*. We note that the definition of a macrocycle differs between *TorchRef*, *phenix.refine* and *REFMAC5*, which is reflected in the

differing convergence behaviour. The compiled Fortran implementation of *REFMAC5* and its use of second-derivative information make it more efficient than either *phenix.refine* or *TorchRef*.

2.4. Performance

TorchRef implements both direct summation-based and FFT-based structure factor calculations. Most processes default to the FFT-based structure factor calculation, as it is more efficient. This calculation proceeds in three stages. First, we sample the model's electron density on a real-space grid in $P1$. This involves sampling the per-atom electron density on a local grid (with a per-atom truncation radius based on the ADP and atomic form factor of the atom) and then coalescing the local grids onto a global grid (*Supplementary methods, FFT path and electron-density sampling*). Second, we perform an FFT on the grid. Last, we extract the structure factors from the grid in $P1$ and apply the symmetry operations (*Supplementary methods, Reciprocal basis and late symmetry expansion*).

Across a set of benchmark structures spanning roughly 1300–10000 atoms we compare *TorchRef*'s FFT-based structure factor calculation against *CCTBX*'s (Grosse-Kunstleve *et al.*, 2002) optimized routine. We include *SF Calculator*'s direct-summation GPU path (Li *et al.*, 2025) in this comparison as it covers similar functionality in enabling structure factor calculations in a *PyTorch* framework. On a single CPU core, *TorchRef*'s FFT-based forward F_{calc} is generally on par with *CCTBX*'s routine [Fig. 3(a)]. At four cores, *TorchRef* is generally faster than *CCTBX*, with further improvement at 16 cores. When running on the GPU (A100) with the custom Triton kernel (Tillet *et al.*, 2019), the forward F_{calc} is a median 124-fold faster than *CCTBX* [52–179-fold across the set; Fig. 3(a)]. For comparison, the direct-summation-based structure factor path of *SF Calculator* (Li *et al.*, 2025) running on an A100 is roughly comparable in performance to *CCTBX*, but suffers from OOM errors on the largest two structures [Fig. 3(a)].

A per-target breakdown for pdb_00001daw (Niefind *et al.*, 1999) of a full refinement iteration shows that the X-ray target dominates the cost [Fig. 3(b)]. While the classic refinement workload cannot fully exploit the speed of GPU structure factor calculations, it does show a significant speedup compared with CPU evaluation. A complete forward-plus-backward cycle takes ~ 14 ms on the GPU and ~ 75 ms on eight CPU cores, roughly a fivefold GPU speedup over CPU evaluation.

2.5. Difference refinement

The end-to-end differentiability of *TorchRef* enables gradient propagation through complex computational pipelines that would be difficult to implement in conventional refinement programs. In particular, it supports multi-model, multi-dataset workflows. As an example, we implement difference-amplitude-based refinement (Terwilliger &

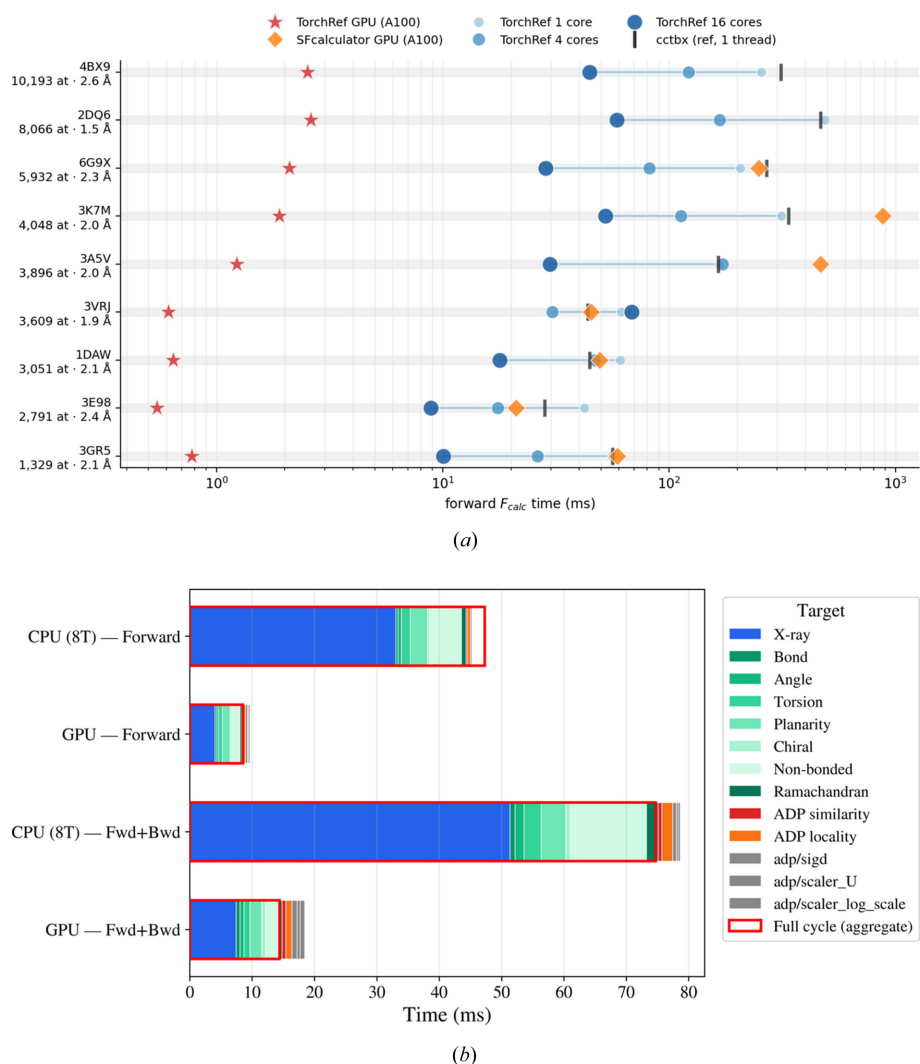


Figure 3

Structure factor and refinement-cycle performance. GPU timings were recorded on an NVIDIA A100 40 GB and CPU and *CCTBX* timings on an Intel Xeon 6230R (Cascade Lake), chosen to be generation-matched to the A100. (a) Forward F_{calc} time across nine benchmark structures (one row each, labelled by atom count and resolution; log scale), comparing *TorchRef* on the GPU (red stars) and on 1/4/16 CPU cores (light-to-dark blue circles) against *SFCalculator* on the GPU (orange diamonds) and *CCTBX* on a single CPU thread (black ticks, reference). *TorchRef* matches *CCTBX* on a single core and overtakes it at four cores; on the GPU it is a median 124-fold faster than *CCTBX* (52–179-fold across the set). *SFCalculator*'s direct-summation GPU path is one to two orders of magnitude slower than *TorchRef*'s GPU kernel and is absent for the two largest structures, where it runs out of memory. (b) Per-target time breakdown of a refinement cycle for pdb_00001daw (3051 atoms, 23356 reflections, $d_{min} = 2.05$ Å) on CPU (eight threads) and GPU, for the forward pass and the full forward + backward pass. Stacked bars show the individually timed targets: the X-ray target dominates, and the hollow red box is the independently measured full cycle. A full forward + backward cycle takes ~14 ms on the GPU and ~75 ms on eight CPU cores.

Berendzen, 1995), which is useful for time-resolved crystallography and related fields. Alongside the geometric restraints and atomic displacement regularization already in the refinement, we add an X-ray difference loss: a Gaussian negative log-likelihood on the amplitude differences between two related datasets (a Gaussian negative log-likelihood is, up to additive constants independent of the refined parameters, equivalent to an inverse-variance-weighted least-squares residual). Because the target models the difference directly, systematic errors common to both datasets cancel rather than bias the fit. We further place a spike-and-slab prior on the per-atom coordinate change between the dark and light models, treating small

displacements as noise (the spike) while leaving larger, genuine movements essentially unpunished (the slab). The full target, the mixed-model parameterization, and the prior are derived in *Supplementary methods, Difference refinement for time-resolved data*.

Combining this with mixed-model refinement, where we define a combined model as a linear combination of a reference and a light model, we can optimize the light model to explain the observed structure factor differences. This is a simple example of a multi-model, multi-dataset approach that can be extended to fit the data (*Supplementary methods, Multi-dataset and multi-model infrastructure*).

To validate different solutions, we implemented a command-line tool that calculates local correlation coeffi-

cients between the weighted difference observed structure factor (WDF_o) density map and the weighted difference calculated structure factor (WDF_c) density map for a given atom selection and masking radius (see *Materials and methods*) under consideration of the dark phases. Here, we use inverse-variance weights for the difference observations. Additionally, we calculate reciprocal-space correlation on the differences. Localization of the signal within the structure yields noticeably higher real-space correlations than reciprocal-space correlations, allowing meaningful correlations to be extracted even for a strongly localized change.

To assess the utility of the difference-refinement approach, we applied it to dataset pdb_00007yyz from a time-resolved serial crystallography study of the photo-pharmacological drug azo-combretastatin A4 (azo-CA4) bound to $\alpha\beta$ -tubulin (Wranik *et al.*, 2023) (for the full refinement protocol, see *Materials and methods*). In that study, *cis*-to-*trans* photo-isomerization of the azobenzene moiety was triggered by light, and nine structural snapshots spanning 1 ns to 100 ms captured the resulting ligand release and collapse of the

colchicine-site binding pocket. The pdb_00007yyz dataset corresponds to the 10 μ s timepoint, at which the ligand has begun to shift but remains bound. The deposited structure was originally refined with *phenix.refine* using extrapolated structure factors at an estimated light-state occupancy of 0.22.

TorchRef difference refinement was performed with a resolution cutoff of 2.2 Å at an assumed light-state occupancy of 0.22, matching the published structure. The occupancy can optionally be refined, but from a single timepoint this is generally not recommended and any refined value must be interpreted with care, so we fixed it to the published estimate. Within a 2.5 Å radius around both the dark and light conformations of the IBL ligand (azo-CA4), the resulting model matched the WDF_o with a local correlation of 0.85, in contrast to 0.74 for the deposited model refined using extrapolated structure factors (Fig. 4). Considering the global fit of the model, the calculated difference structure factor amplitudes show correlations of 0.51 and 0.31 with the observed difference structure factor amplitudes for the *TorchRef* difference-refined model and the extrapolated model, respectively.

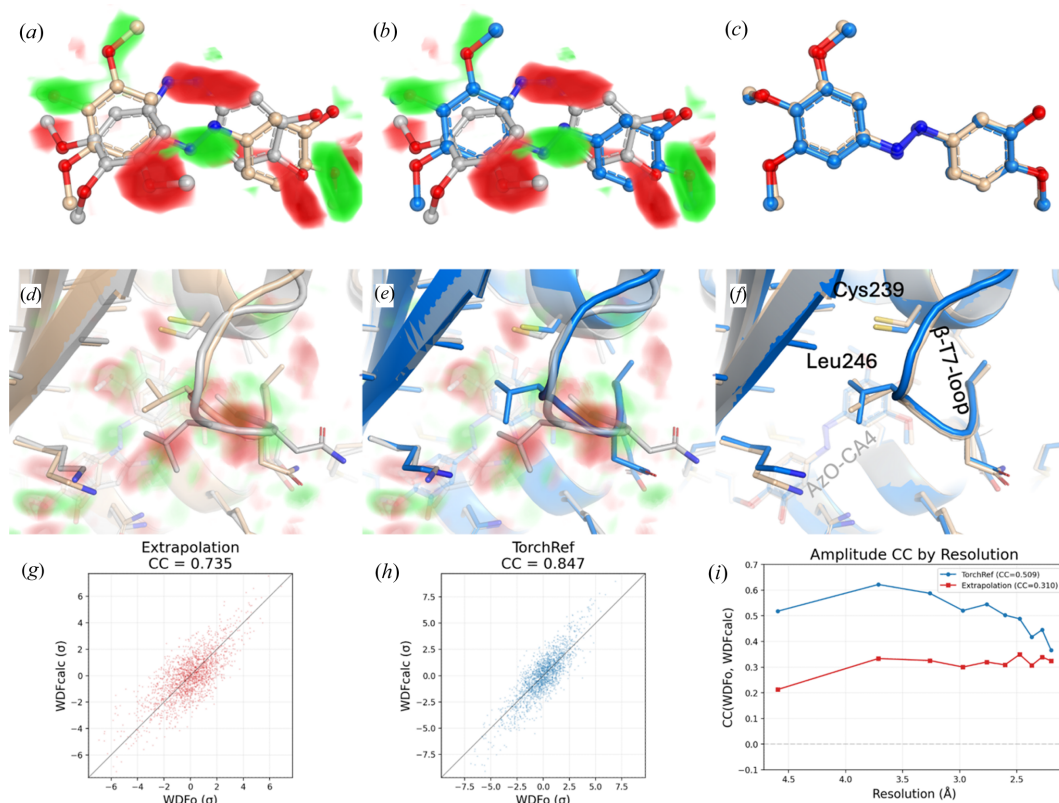


Figure 4

Difference refinement of azo-CA4 (IBL) *cis*-to-*trans* isomerization in tubulin (pdb_00007yyz). Weighted difference electron-density (WDF_o) maps are visualized via a volume ramp from 3 to 5σ around the ligand (green, positive; red, negative) (*a*, *b*) and 2.5 to 4.5σ in the wider binding pocket (*d*, *e*). Top row (*a*–*c*): IBL ligand close-up. Bottom row (*d*–*f*): binding pocket view showing the changes in the β 7-loop resulting from the unbinding of the *trans* conformer ligand. Dark-state structure pdb_00008ql2 (grey), published extrapolated structure pdb_00007yyz (wheat) and *TorchRef* difference-refined structure (blue). (*a*, *d*) WDF_o map with the deposited pdb_00007yyz structure refined from extrapolated structure factors. (*b*, *e*) WDF_o map with the *TorchRef* difference-refined structure. (*c*, *f*) Superposition of the *TorchRef*-refined (blue) and deposited pdb_00007yyz (wheat) conformations, showing the agreement and differences between the independently determined light-state structures. Scatter plots (*g*) and (*h*) show the distribution of WDF_c voxel values relative to WDF_o voxels within a 2.5 Å radius around the dark and light conformations of the ligand. (*g*) shows the scatter plot for the extrapolated data, while (*h*) shows the *TorchRef* difference-refinement scatter plot. The correlation between calculated and observed difference amplitudes is shown in panel (*i*), binned by resolution.

Notably, the differences in the resulting structures are subtle and the models are qualitatively similar.

3. Discussion

TorchRef provides a *PyTorch*-based crystallographic refinement framework that exposes all relevant parameters to automatic differentiation. This design paradigm offers significant advantages over classic implementations, as it correctly handles complex gradient computations through automatic backpropagation, removing the need for manually derived analytical gradients. This not only reduces implementation effort but also eliminates an important source of programming errors when developing new refinement targets or parameterizations. *TorchRef*'s native *PyTorch* implementation enables deep integration within the growing ecosystem of *PyTorch*-based tools in structural biology, including molecular-dynamics engines such as *TorchMD* (Doerr *et al.*, 2021), structural prediction models such as *OpenFold* (Ahdritz *et al.*, 2024) [as demonstrated by the recent Rocket project (Fadini *et al.*, 2026)] and machine-learning-accelerated quantum-mechanical potentials such as those used in *AQuaRef* (Zubatyuk *et al.*, 2025).

A central concern for any new refinement framework is whether its design choices compromise refinement quality. Across 725 *Phaser*-placed *AlphaFold* predictions refined head-to-head with *phenix.refine*, *REFMAC5* and *TorchRef*, the three engines reach median R_{free} values within 0.30 percentage points of each other (*TorchRef* 0.3165, *phenix.refine* 0.3161, *REFMAC5* 0.3135), a spread small enough that the differences carry no practical weight, even though the paired comparison against *REFMAC5* remains statistically resolvable in its favour. *TorchRef* reaches the lowest median R_{work} but the largest $R_{\text{work}}-R_{\text{free}}$ gap (0.0470, against 0.0416–0.0417 for the reference engines), a mild overfitting signature. We hypothesize that the current global fixed-weight target contributes to it and that resolution-dependent or adaptive weighting would reduce it, though the concurrent tendency towards low absolute ADPs (main-chain bonded- B RMSZ 1.55 against 0.90 for *REFMAC5*) leaves open an ADP-driven component. *TorchRef*'s coordinate-geometry RMSZ is essentially on par with *REFMAC5*, consistent with the use of the same CCP4 Monomer Library restraint definitions. The one clear residual is a small R_{free} excess relative to *REFMAC5* (paired median +0.0030), whereas against *phenix.refine* *TorchRef* is marginally lower.

The difference-refinement procedure demonstrates a use case that is difficult to implement in conventional refinement programs. Existing approaches to time-resolved crystallographic data analysis typically rely on extrapolated structure factors, which amplify the signal from a partially occupied excited state to simulate a fully occupied one. While powerful, extrapolation introduces noise amplification that scales inversely with occupancy.

Additionally, because extrapolated structure factor amplitudes are calculated with phases derived from the dark state,

any phase difference between the dark and light models introduces error into the resulting maps. Extrapolation also systematically produces negative amplitudes from phase mismatches and noise amplification. The standard correction protocols absorb this as additional noise in the active region rather than as a systematic bias, which accounts for the general success of extrapolation-based methods while leaving the excited-state density noisier than the true signal. Modern extrapolation tools further apply Bayesian weighting that downweights noisy-difference amplitudes, at the cost of shrinking the signal towards the dark reference (De Zitter *et al.*, 2022).

Direct refinement of a mixed model against amplitude differences, as implemented in *TorchRef*, sidesteps these issues by fitting the difference between the dark-state model and the occupancy-weighted combination of dark- and light-state models to the observed difference in the dark and light data. In the case of pdb_00007yyz, this approach yielded a higher local correlation between calculated and observed difference maps (0.85 versus 0.74 for the deposited structure around the chromophore). The differences between the extrapolated structure and the *TorchRef* difference-refined structure are subtle, but the local-correlation improvement is reproduced across the chromophore region. Beyond the two-dataset case shown here, this approach extends naturally to more than two datasets and multiple transient structures.

TorchRef is designed as a platform for method development and does not yet implement the full feature set of established refinement packages. On the modelling side, it lacks automated water placement and deletion, non-crystallographic symmetry restraints, reference-model restraints, twinning support and translation–libration–screw parameterization. On the optimization side, simulated annealing and Langevin-style stochastic optimizers are implemented as experimental alternatives, but they are not part of the production protocol benchmarked here and still require systematic validation. On the target side, adaptive weighting remains an open problem and may help reduce the small residual R_{free} excess. Qualitative inspection of the rerun models shows a systematic tendency towards low absolute ADPs, pointing to a possible scale-factor–ADP interaction; this diagnosis is provisional and will be tested in future work.

Looking ahead, *TorchRef*'s modular architecture opens several avenues for development. The multi-model, multi-dataset framework demonstrated here for difference refinement generalizes naturally to kinetic multi-state refinement against time-resolved serial crystallography data, where a small number of conformational states would be refined against many datasets simultaneously. As an experimental proof of concept for force-field integration, *TorchRef* already includes a working differentiable AMBER14/GAFF2 target (Maier *et al.*, 2015; Wang *et al.*, 2004) coupled to OpenMM (Eastman *et al.*, 2017), with analytical forces bridged into *PyTorch* autograd, non-standard residues parameterized on the fly via *antechamber* (Wang *et al.*, 2006) and a riding-hydrogen model that redistributes H forces onto the heavy-atom frame. The AMBER target is not yet at production

quality and is not used in the *AlphaFold* benchmark above. Replacing the empirical AMBER potential with learned energy surfaces such as MACE-OFF (Kovács *et al.*, 2025) or AIMNet2 (Anstine *et al.*, 2025) would combine deep-learned physical priors with the data-driven X-ray maximum likelihood, much as *AQuaRef* (Zubatyuk *et al.*, 2025) has demonstrated for quantum-mechanical refinement within *Phenix*. However, the advantage is that, in *TorchRef*, the potential is a native *PyTorch* module and its gradients propagate through the same computational graph as the other targets of the refinement.

Given the well documented, open-source nature of *TorchRef*, we hope to make the development of custom refinement protocols accessible to a broader community of crystallographers and method developers. The full API documentation, Jupyter notebook tutorials and permissive MIT licence are intended to lower the barrier to experimentation. As a concrete example, the difference refinement was initially implemented in approximately 1000 lines of Python code (of which the core logic accounts for 200 lines), a task that would require substantially more effort in a conventional refinement codebase, where analytical gradients must be derived and implemented by hand.

We hope that *TorchRef*, or projects like it, will allow many more researchers to contribute effectively to macromolecular structure refinement, a field in which the depth and complexity of established codebases have historically concentrated methodological development among a small number of specialists. We are open to contributions and collaborations, and welcome feedback to help shape the future of *TorchRef*.

4. Materials and methods

4.1. Structure selection and dataset preparation

A non-redundant benchmark dataset of 1000 protein crystal structures was assembled from the PDB. Structures were selected programmatically via the RCSB PDB Search API using the following criteria: (i) experimental method restricted to X-ray diffraction, (ii) resolution between 1.5 and 3.0 Å, and (iii) maximum pairwise sequence identity of 30%, enforced by retaining only cluster representatives from the RCSB sequence clustering at the 30% identity level. From an initial pool of 1500 candidate entries, structures were filtered to retain only those with deposited structure factor data, yielding the final set of 1000 structures. For each structure, coordinate files [PDB and macromolecular CIF (mmCIF) formats] and structure factor files [structure factor CIF (SF-CIF) format] were downloaded from the RCSB.

4.2. MTZ conversion and R_{free} flag generation

SF-CIF files were converted to MTZ format using *TorchRef*'s *ReflectionData* module. R_{free} flags were regenerated for all structures [`regenerate_rfree_flags(force=True)`], as deposited flags were inconsistent across the benchmark set. Flags are assigned by resolution-stratified random sampling: the reflections are binned in

shells of d^{*2} and a fixed fraction of each shell is drawn at random. Reflections are stored in the canonical CCP4 asymmetric unit, so every space-group-symmetry equivalent maps to a single asymmetric unit reflection and shares its flag. The generator does not, however, currently group reflections related by higher metric (lattice) symmetry (for example, twin operators or pseudosymmetry not expressed by the space group), so under twinning or pseudosymmetry a free reflection could be related to a working reflection, leaking information across the work/free boundary; metric-symmetry-aware flag grouping is planned as future work. We did not perform a dedicated cohort-wide twinning or pseudosymmetry screen and therefore do not claim that these pathologies are absent from every benchmark structure.

4.3. *AlphaFold* search model preparation and molecular replacement

For each benchmark structure, each protein chain was mapped to its UniProt accession via the RCSB Data API (GraphQL endpoint), and the corresponding *AlphaFold2* model was downloaded from the *AlphaFold* Protein Structure Database (Varadi *et al.*, 2022; <https://alphafold.ebi.ac.uk/api/prediction/accession>). Structures for which every protein chain had an *AlphaFold* model (`af_complete`) defined the *AlphaFold* benchmark cohort.

For each UniProt component, the *AlphaFold* model was aligned to the crystallized construct sequence using *gemmi* pairwise alignment (Wojdyr, 2022). From the alignment (i) *AlphaFold*-only flanks and insertions were trimmed; (ii) residues matching the construct were retained; and (iii) residues whose *AlphaFold* identity disagreed with the construct were truncated to alanine (residue renamed ALA, side chain stripped to {N, C α , C, O, C β }), except proline and glycine, which were kept as is. Trimmed search models were then pLDDT $\rightarrow$ B-converted with `phenix.process_predicted_model` (`maximum_rmsd=100`, disabling confidence-based residue removal since the model was already trimmed) and placed by *phenix.phaser* in MR_AUTO mode using the crystallized composition (ENSEMBLE ... IDENTITY 0.9; one SEARCH ENSEMBLE ... NUMBER {copies} per UniProt component). The top *Phaser* solution was retained as the placed search model. Out of 825 *AlphaFold*-complete entries, *Phaser* produced a solution for 767 structures (93%), defining the benchmark arm used here. The 58 structures that failed *Phaser* MR were excluded.

4.4. Refinement protocols

The placed *AlphaFold* model was refined against the structure factors by each engine in turn, using matched protocols and identical hardware (Slurm-managed cluster, four CPU cores and 8 GB of RAM per job):

TorchRef: `torchref.refine` (v0.6.4), 10 macrocycles, alternating coordinate, ADP/occupancy and scaler blocks (`--mode separate`); the default σ_A maximum-likelihood X-ray target (`--X-ray-mode ml`) and default group weights $w_{\text{xray}} = 1.0$, $w_{\text{geometry}} = 0.2$, $w_{\text{adp}} = 0.02$.

Phenix: *phenix.refine* v1.20-4459 (Liebschner *et al.*, 2019), 10 macrocycles with the `individual_sites + individual_adp + occupancies` strategy (separate refinement of atomic coordinates, individual isotropic *B* values and partial occupancies), bulk-solvent and overall scaling enabled, ordered solvent disabled, Ramachandran restraints enabled, simulated annealing and target-weight optimization disabled.

REFMAC5: *refmac5* from the CCP4 suite v7.1 (Murshudov *et al.*, 2011; Agirre *et al.*, 2023), 10 NCYCLES restrained refinement, MAKE HYDR NO, with `FreeR_flag` read from the input MTZ.

4.5. *R*-factor scoring and cross-scorer validation

All refined models, and the unrefined placed search models for reference, were independently re-scored against identical structure factors and free flags by three *R*-factor scorers operating in zero-cycle mode (scaling and bulk-solvent fit only; coordinates and ADPs untouched): *REFMAC5* with NCYCLES 0 (Murshudov *et al.*, 2011); *phenix.model_vs_data* (Liebschner *et al.*, 2019); and *torchref.refine -n 0* on the same model. For the main figure, *phenix.model_vs_data* provides the headline *R* factors. Geometry RMSZ values reported in Fig. 2(b) were computed from the *REFMAC5* zero-cycle restraint table so the underlying restraint library is uniform across engines. Wall-clock per refinement was taken from each program's own log files for successful runs.

4.6. Benchmarking structure factor calculation and gradient performance

4.6.1. Test system

The forward F_{calc} scaling benchmark [Fig. 3(a)] used a set of nine structures spanning roughly 1300–10000 atoms and resolutions of 1.5–2.6 Å. The per-target refinement-cycle breakdown [Fig. 3(b)] used the crystal structure `pdb_00001daw` (Niefind *et al.*, 1999), comprising 3051 atoms and 23356 unique reflections at a resolution of 2.05 Å.

4.6.2. Quantities benchmarked

Three timing scenarios were measured for *TorchRef*, each calling `compute_structure_factors` directly (bypassing the model cache):

(1) Forward: structure factor calculation $F_{\text{calc}}(\mathbf{h})$ inside a `torch.no_grad` context, measuring the raw forward-pass cost without constructing a computational graph.

(2) Backward only: a forward pass followed by `loss.backward()` (where $\text{loss} = \sum |F_{\text{calc}}|$), with only the backward pass timed. This isolates the cost of automatic differentiation through the structure factor kernel. This timing is less accurate than combined forward + backward pass evaluations due to timing overhead.

(3) Forward + backward: end-to-end F_{calc} computation plus gradient backpropagation, timed together.

As a reference, the forward-only F_{calc} computation via *CCTBX* [`xray_structure.structure_factors(d_min=...).f_calc()`] was timed on a CPU under identical conditions, using *cctbx-base* (the version pinned by the *TorchRef* installation) and the same `OMP_NUM_THREADS/MKL_NUM_THREADS/OPENBLAS_NUM_THREADS` caps as the *TorchRef* runs. *CCTBX* gradients were not benchmarked: a documented public API for differentiating `f_calc()` through *CCTBX* was not identified at the time of writing. As a second reference, *SFCalculator* (Li *et al.*, 2025) was benchmarked on the GPU for the forward F_{calc} under matching conditions; OOM failures on the largest structures were recorded as such.

4.6.3. CPU thread scaling

To assess parallelism, benchmarks were run across one to 16 CPU threads; the F_{calc} scaling figure reports one, four and 16 cores. Each configuration was executed in an isolated subprocess with `TORCHREF_NUM_THREADS`, `OMP_NUM_THREADS`, `MKL_NUM_THREADS` and `OPENBLAS_NUM_THREADS` all set to the target thread count, ensuring consistent control over intra-operator parallelism in *PyTorch*, BLAS and OpenMP. The *TorchRef* `__init__` module reads `TORCHREF_NUM_THREADS` and calls `torch.set_num_threads()` accordingly.

4.6.4. GPU benchmarks

GPU timings were collected on an NVIDIA A100-PCIE-40 GB. All GPU timings bracket each iteration with `torch.cuda.synchronize()` to ensure asynchronous kernel launches are fully completed before reading the wall-clock time.

4.6.5. Timing protocol

For each configuration (thread count and device), three warm-up iterations were run followed by ten timed iterations. Wall-clock time was measured with `time.perf_counter()`. Summary statistics reported are the mean, minimum and maximum over the ten timed iterations. Speedup is defined as the ratio of the single-thread CPU mean time to the mean time at a given thread count (or on a GPU).

4.6.6. Loss function timing

Loss function timings used *TorchRef*'s default loss function initialization, as in a standard refinement setup. We then evaluated the forward and backward passes for the loss functions commonly used in refinement.

4.7. Difference refinement

Difference refinement was performed with *torchref.difference-refine*, which targets the residual between the observed and calculated structure factor amplitude differences of the light and dark datasets, with the dark model held fixed (see *Supplementary methods, Difference refinement for time-resolved data*). Refined models were validated with

`torchref.validate-ded`, which reports the local correlation between the weighted observed difference map (WDF_{obs} , from $F_{\text{obs}}^{\text{light}} - F_{\text{obs}}^{\text{dark}}$) and the weighted calculated difference map (WDF_{c} , from $|F_{\text{calc}}^{\text{mix}}| - |F_{\text{calc}}^{\text{dark}}|$) within a masking radius around a selected atom set; maps were calculated using dark-state phases and Bayesian weights.

To demonstrate the approach we refined against PDB entry `pdb_00007yyz` (Wranik *et al.*, 2023), a time-resolved serial crystallography structure of a photoisomerized azobenzene-analogue ligand. The deposited model was originally refined with *phenix.refine* using extrapolated structure factors at 0.22 occupancy and a 2.2 Å resolution cutoff. For *TorchRef* difference refinement we used the recently released dark structure `pdb_00008ql2` (Weinert, T., Wranik, M., Seidel, H.-P., Church, J., Steinmetz, M.O., Schapiro, I. & Standfuss, J., to be published) as both the initial light and dark model with a fixed 22% light-state fraction. Following an automated pass, two cycles of manual real-space adjustment in *Coot* (Emsley *et al.*, 2010), using the phase-corrected extrapolated structure factors written by `torchref.difference-refine`, alternated with refinement; a final equilibration cycle set both the difference-target and coordinate-similarity weights to 1. Reflections were used to 2.2 Å, validation correlations were computed at the same resolution to allow direct comparison with the deposited model, and R_{free} flags were grafted from the `pdb_00008ql2` dataset to give meaningful R_{free} values.

5. Related literature

The following references are cited in the supporting information: Brünger (1992), Croll & Read (2021), Cromer & Liberman (1981), Engh & Huber (1991), Lunin & Skovoroda (1995), Masmaliyeva *et al.* (2020), Mitchell & Beauchamp (1988), Nicholls *et al.* (2014), Sheldrick (2015), Wilson (1992).

6. Data and code availability

TorchRef is available on GitHub under an MIT licence and can be installed via *pip*. Full API documentation and *Jupyter* notebook tutorials are available via GitHub at <https://github.com/HatPdotS/TorchRef>. The scripts used to generate the figures are available via GitHub at <https://github.com/HatPdotS/TorchRef-paper>. The difference-refined structure is pending release in the PDB and will be available under code `pdb_0000030as`.

Acknowledgements

The project was conceived by H-PS and TW. The code was authored by H-PS under the supervision of TW and JS. The manuscript was authored by H-PS and TW with input from JS. Figures were made by H-PS with the help of TW. The manuscript was reviewed by JS, TW and H-PS. Claude Code was used during the development of *TorchRef* for prototyping, refactoring and documentation. Claude was used to review and rephrase text during the preparation of this manuscript. Open access publishing facilitated by ETH-Bereich

Forschungsanstalten, as part of the Wiley–ETH-Bereich Forschungsanstalten agreement via the Consortium Of Swiss Academic Libraries.

Conflict of interest

The authors declare that they have no conflicts of interest.

Funding information

This work was supported by Swiss National Science Foundation project grants 10.007.643 (JS) and 10.003.847 (TW).

References

- Adams, P. D., Afonine, P. V., Bunkóczi, G., Chen, V. B., Davis, I. W., Echols, N., Headd, J. J., Hung, L.-W., Kapral, G. J., Grosse-Kunstleve, R. W., McCoy, A. J., Moriarty, N. W., Oeffner, R., Read, R. J., Richardson, D. C., Richardson, J. S., Terwilliger, T. C. & Zwart, P. H. (2010). *Acta Cryst.* **D66**, 213–221.
- Afonine, P. V., Grosse-Kunstleve, R. W. & Adams, P. D. (2005). *Acta Cryst.* **D61**, 850–855.
- Afonine, P. V., Grosse-Kunstleve, R. W., Echols, N., Headd, J. J., Moriarty, N. W., Mustyakimov, M., Terwilliger, T. C., Urzhumtsev, A., Zwart, P. H. & Adams, P. D. (2012). *Acta Cryst.* **D68**, 352–367.
- Agirre, J., Atanasova, M., Bagdonas, H., Ballard, C. B., Baslé, A., Beilsten-Edmands, J., Borges, R. J., Brown, D. G., Burgos-Mármol, J. J., Berrisford, J. M., Bond, P. S., Caballero, I., Catapano, L., Chojnowski, G., Cook, A. G., Cowtan, K. D., Croll, T. I., Debreczeni, J. É., Devenish, N. E., Dodson, E. J., Drevon, T. R., Emsley, P., Evans, G., Evans, P. R., Fando, M., Foadi, J., Fuentes-Montero, L., Garman, E. F., Gerstel, M., Gildea, R. J., Hatti, K., Hekkelman, M. L., Heuser, P., Hoh, S. W., Hough, M. A., Jenkins, H. T., Jiménez, E., Joosten, R. P., Keegan, R. M., Keep, N., Krissinel, E. B., Kolenko, P., Kovalevskiy, O., Lamzin, V. S., Lawson, D. M., Lebedev, A. A., Leslie, A. G. W., Lohkamp, B., Long, F., Malý, M., McCoy, A. J., McNicholas, S. J., Medina, A., Millán, C., Murray, J. W., Murshudov, G. N., Nicholls, R. A., Noble, M. E. M., Oeffner, R., Pannu, N. S., Parkhurst, J. M., Pearce, N., Pereira, J., Perrakis, A., Powell, H. R., Read, R. J., Rigden, D. J., Rochira, W., Sammito, M., Sánchez Rodríguez, F., Sheldrick, G. M., Shelley, K. L., Simkovic, F., Simpkin, A. J., Skubak, P., Sobolev, E., Steiner, R. A., Stevenson, K., Tews, I., Thomas, J. M. H., Thorn, A., Valls, J. T., Uski, V., Usón, I., Vagin, A., Velankar, S., Vollmar, M., Walden, H., Waterman, D., Wilson, K. S., Winn, M. D., Winter, G., Wojdyr, M. & Yamashita, K. (2023). *Acta Cryst.* **D79**, 449–461.
- Ahdritz, G., Bouatta, N., Floristean, C., Kadyan, S., Xia, Q., Gerecke, W., O'Donnell, T. J., Berenberg, D., Fisk, I., Zanichelli, N., Zhang, B., Nowaczynski, A., Wang, B., Stepniowska-Dziubinska, M. M., Zhang, S., Ojewole, A., Guney, M. E., Biderman, S., Watkins, A. M., Ra, S., Lorenzo, P. R., Nivon, L., Weitzner, B., Ban, Y. A., Chen, S., Zhang, M., Li, C., Song, S. L., He, Y., Sorger, P. K., Mostaque, E., Zhang, Z., Bonneau, R. & AlQuraishi, M. (2024). *Nat. Methods* **21**, 1514–1524.
- AlQuraishi, M. & Sorger, P. K. (2021). *Nat. Methods* **18**, 1169–1180.
- Anstine, D. M., Zubatyuk, R. & Isayev, O. (2025). *Chem. Sci.* **16**, 10228–10244.
- Barends, T. R. M., Stauch, B., Cherezov, V. & Schlichting, I. (2022). *Nat. Rev. Methods Primers* **2**, 59.
- Blanc, E., Roversi, P., Vornrhein, C., Flensburg, C., Lea, S. M. & Bricogne, G. (2004). *Acta Cryst.* **D60**, 2210–2221.
- Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Katariya, Y., Leary, C., Maclaurin, D., Nacula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S. & Zhang, Q. (2018). *JAX: Composable*

- Transformations of Python+NumPy Programs*, <https://github.com/jax-ml/jax>.
- Brändén, G. & Neutze, R. (2021). *Science* **373**, eaba0954.
- Brünger, A. T. (1992). *Nature* **355**, 472–475.
- Burley, S. K., Berman, H. M., Bhikadiya, C., Bi, C., Chen, L., Di Costanzo, L., Christie, C., Dalenberg, K., Duarte, J. M., Dutta, S., Feng, Z., Ghosh, S., Goodsell, D. S., Green, R. K., Guranović, V., Guzenko, D., Hudson, B. P., Kalro, T., Liang, Y., Lowe, R., Namkoong, H., Peisach, E., Periskova, I., Prlić, A., Randle, C., Rose, A., Rose, P., Sala, R., Sekharan, M., Shao, C., Tan, L., Tao, Y.-P., Valasatava, Y., Voigt, M., Westbrook, J., Woo, J., Yang, H., Young, J., Zhuravleva, M. & Zardecki, C. (2019). *Nucleic Acids Res.* **47**, D464–D474.
- Chapman, H. N. (2019). *Annu. Rev. Biochem.* **88**, 35–58.
- Clabbers, M. T. B. & Xu, H. (2021). *Acta Cryst.* **D77**, 313–324.
- Croll, T. I. & Read, R. J. (2021). *Acta Cryst.* **D77**, 438–446.
- Cromer, D. T. & Liberman, D. A. (1981). *Acta Cryst.* **A37**, 267–268.
- De Zitter, E., Coquelle, N., Oeser, P., Barends, T. R. M. & Colletier, J.-P. (2022). *Commun. Biol.* **5**, 640.
- Doerr, S., Majewski, M., Pérez, A., Krämer, A., Clementi, C., Noe, F., Giorgino, T. & De Fabritiis, G. (2021). *J. Chem. Theory Comput.* **17**, 2355–2363.
- Eastman, P., Swails, J., Chodera, J. D., McGibbon, R. T., Zhao, Y., Beauchamp, K. A., Wang, L.-P., Simonnet, A. C., Harrigan, M. P., Stern, C. D., Wiewiora, R. P., Brooks, B. R. & Pande, V. S. (2017). *PLoS Comput. Biol.* **13**, e1005659.
- Emsley, P., Lohkamp, B., Scott, W. G. & Cowtan, K. (2010). *Acta Cryst.* **D66**, 486–501.
- Engl, R. A. & Huber, R. (1991). *Acta Cryst.* **A47**, 392–400.
- Fadini, A., Apostolopoulou, V., Lane, T. J. & van Thor, J. J. (2025). *Commun. Biol.* **8**, 1649.
- Fadini, A., Li, M., McCoy, A. J., Banjara, S., Okumura, H., Napier, E., Fontana, P., Khan, A. R., Jovine, L., Terwilliger, T. C., Read, R. J., Hekstra, D. R. & AlQuraishi, M. (2026). *Nat. Methods* **23**, 785–795.
- French, S. & Wilson, K. (1978). *Acta Cryst.* **A34**, 517–525.
- Gasparotto, P., Barba, L., Stadler, H.-C., Assmann, G., Mendonça, H., Ashton, A. W., Janousch, M., Leonarski, F. & Béjar, B. (2024). *J. Appl. Cryst.* **57**, 931–944.
- Greisman, J. B., Dalton, K. M. & Hekstra, D. R. (2021). *J. Appl. Cryst.* **54**, 1521–1529.
- Grosse-Kunstleve, R. W., Sauter, N. K., Moriarty, N. W. & Adams, P. D. (2002). *J. Appl. Cryst.* **35**, 126–136.
- Hekstra, D. R. (2023). *Annu. Rev. Biophys.* **52**, 255–274.
- Henning Hansen, C. G., Orioli, S. & Lindorff-Larsen, K. (2026). *ACS Phys. Chem. Au* **6**, 878–886.
- Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., Tunyasuvunakool, K., Bates, R., Židek, A., Potapenko, A., Bridgland, A., Meyer, C., Kohl, S. A. A., Ballard, A. J., Cowie, A., Romera-Paredes, B., Nikolov, S., Jain, R., Adler, J., Back, T., Petersen, S., Reiman, D., Clancy, E., Zielinski, M., Steinegger, M., Pacholska, M., Berghammer, T., Bodenstein, S., Silver, D., Vinyals, O., Senior, A. W., Kavukcuoglu, K., Kohli, P. & Hassabis, D. (2021). *Nature* **596**, 583–589.
- Khusainov, G., Standfuss, J. & Weinert, T. (2024). *Struct. Dyn.* **11**, 020901.
- Kovács, D. P., Moore, J. H., Browning, N. J., Batatia, I., Horton, J. T., Pu, Y., Kapil, V., Witt, W. C., Magdău, I.-B., Cole, D. J. & Csányi, G. (2025). *J. Am. Chem. Soc.* **147**, 17598–17611.
- Li, M., Dalton, K. & Hekstra, D. (2025). *bioRxiv*, <https://www.biorxiv.org/content/early/2025/01/19/2025.01.12.632630>.
- Liebschner, D., Afonine, P. V., Baker, M. L., Bunkóczi, G., Chen, V. B., Croll, T. I., Hintze, B., Hung, L.-W., Jain, S., McCoy, A. J., Moriarty, N. W., Oeffner, R. D., Poon, B. K., Prisant, M. G., Read, R. J., Richardson, J. S., Richardson, D. C., Sammito, M. D., Sobolev, O. V., Stockwell, D. H., Terwilliger, T. C., Urzhumtsev, A. G., Videau, L. L., Williams, C. J. & Adams, P. D. (2019). *Acta Cryst.* **D75**, 861–877.
- Long, F., Nicholls, R. A., Emsley, P., Gražulis, S., Merkys, A., Vaitkus, A. & Murshudov, G. N. (2017). *Acta Cryst.* **D73**, 112–122.
- Lundgren, K. J. M., Calderaru, O., Oksanen, E. & Ryde, U. (2024). *IUCrJ* **11**, 921–937.
- Lunin, V. Yu. & Skovoroda, T. P. (1995). *Acta Cryst.* **A51**, 880–887.
- Maier, J. A., Martinez, C., Kasavajhala, K., Wickstrom, L., Hauser, K. E. & Simmerling, C. (2015). *J. Chem. Theory Comput.* **11**, 3696–3713.
- Masmaliyeva, R. C., Babai, K. H. & Murshudov, G. N. (2020). *Acta Cryst.* **D76**, 926–937.
- McCoy, A. J., Grosse-Kunstleve, R. W., Adams, P. D., Winn, M. D., Storoni, L. C. & Read, R. J. (2007). *J. Appl. Cryst.* **40**, 658–674.
- Mikhailovskii, O., Izmailov, S. A., Xue, Y., Case, D. A. & Skrynnikov, N. R. (2024). *J. Chem. Inf. Model.* **64**, 18–25.
- Mitchell, T. J. & Beauchamp, J. J. (1988). *J. Am. Stat. Assoc.* **83**, 1023–1032.
- Moriarty, N. W., Janowski, P. A., Swails, J. M., Nguyen, H., Richardson, J. S., Case, D. A. & Adams, P. D. (2020). *Acta Cryst.* **D76**, 51–62.
- Murshudov, G. N., Skubák, P., Lebedev, A. A., Pannu, N. S., Steiner, R. A., Nicholls, R. A., Winn, M. D., Long, F. & Vagin, A. A. (2011). *Acta Cryst.* **D67**, 355–367.
- Murshudov, G. N., Vagin, A. A. & Dodson, E. J. (1997). *Acta Cryst.* **D53**, 240–255.
- Nannenga, B. L. & Gonen, T. (2019). *Nat. Methods* **16**, 369–379.
- Niefind, K., Pütter, M., Guerra, B., Issinger, O. G. & Schomburg, D. (1999). *Nat. Struct. Biol.* **6**, 1100–1103.
- Nicholls, R. A., Fischer, M., McNicholas, S. & Murshudov, G. N. (2014). *Acta Cryst.* **D70**, 2487–2499.
- Pannu, N. S. & Read, R. J. (1996). *Acta Cryst.* **A52**, 659–668.
- Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J. & Chintala, S. (2019). *arXiv*, 1912.01703.
- Read, R. J. (1986). *Acta Cryst.* **A42**, 140–149.
- Schlichting, I. (2015). *IUCrJ* **2**, 246–255.
- Schulz, E. C., Yorke, B. A., Pearson, A. R. & Mehrabi, P. (2022). *Acta Cryst.* **D78**, 14–29.
- Sheldrick, G. M. (2015). *Acta Cryst.* **C71**, 3–8.
- Terwilliger, T. C. & Berendzen, J. (1995). *Acta Cryst.* **D51**, 609–618.
- Tillet, P., Kung, H. T. & Cox, D. (2019). *Proceedings of the 3rd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages (MAPL 2019)*, pp. 10–19. New York: ACM.
- Turk, D. (2013). *Acta Cryst.* **D69**, 1342–1357.
- Varadi, M., Anyango, S., Deshpande, M., Nair, S., Natassia, C., Yordanova, G., Yuan, D., Stroe, O., Wood, G., Laydon, A., Židek, A., Green, T., Tunyasuvunakool, K., Petersen, S., Jumper, J., Clancy, E., Green, R., Vora, A., Lutfi, M., Figurnov, M., Cowie, A., Hobbs, N., Kohli, P., Kleywegt, G., Birney, E., Hassabis, D. & Velankar, S. (2022). *Nucleic Acids Res.* **50**, D439–D444.
- Wang, J., Wang, W., Kollman, P. A. & Case, D. A. (2006). *J. Mol. Graphics Modell.* **25**, 247–260.
- Wang, J., Wolf, R. M., Caldwell, J. W., Kollman, P. A. & Case, D. A. (2004). *J. Comput. Chem.* **25**, 1157–1174.
- Wickstrand, C., Katona, G., Nakane, T., Nogly, P., Standfuss, J., Nango, E. & Neutze, R. (2020). *Struct. Dyn.* **7**, 024701.
- Wilson, A. J. C. (1992). Editor. *International Tables for Crystallography*, Vol. C, *Mathematical, Physical and Chemical Tables*. Dordrecht: Kluwer Academic Publishers.
- Wojdyr, M. (2022). *J. Open Source Softw.* **7**, 4200.
- Wranik, M., Weinert, T., Slavov, C., Masini, T., Furrer, A., Gaillard, N., Gioia, D., Ferrarotti, M., James, D., Glover, H., Carrillo, M., Kekilli, D., Stipp, R., Skopintsev, P., Brünle, S., Mühlethaler, T., Beale, J., Gashi, D., Nass, K., Ozerov, D., Johnson, P. J. M., Cirelli, C., Bacellar, C., Braun, M., Wang, M., Dworkowski, F., Milne, C., Cavalli, A., Wachtveitl, J., Steinmetz, M. O. & Standfuss, J. (2023). *Nat. Commun.* **14**, 903.