Frontiers in Physics

TYPE Review
PUBLISHED 28 November 2025
DOI 10.3389/fphy.2025.1711356

Check for updates

OPEN ACCESS

EDITED BY
Vlasis G. Mavrantzas,
University of Patras, Greece

REVIEWED BY
Zedong Peng,
University of Montana, United States
Xindi Gong,
The University of Texas at Austin, United States

*CORRESPONDENCE
Nikolaos Cheimarios,
nixeimar@chemeng.ntua.gr

[†]PRESENT ADDRESS
Nikolaos Cheimarios,
Accenture, Athens, Greece

RECEIVED 23 September 2025
REVISED 26 October 2025
ACCEPTED 11 November 2025
PUBLISHED 28 November 2025

CITATION
Cheimarios N (2025) Scientific software development in the AI era: reproducibility, MLOps, and applications in soft matter physics.
Front. Phys. 13:1711356.
doi: 10.3389/fphy.2025.1711356

Scientific software development in the AI era: reproducibility, MLOps, and applications in soft matter physics

Nikolaos Cheimarios*[†]

School of Chemical Engineering, National Technical University of Athens, Athens, Greece

Artificial intelligence (AI) is redefining the foundations of scientific software development by turning once-static codes into dynamic, data-dependent systems that require continuous retraining, monitoring, and governance. This article offers a practitioner-oriented synthesis for building reproducible, sustainable, and trustworthy scientific software in the AI era, with a focus on soft matter physics as a demanding yet fertile proving ground. We examine advances in machine-learned interatomic and coarse-grained potentials, differentiable simulation engines, and closed-loop inverse design strategies, emphasizing how these methods transform modeling workflows from exploratory simulations into adaptive, end-to-end pipelines. Drawing from software engineering and MLOps, we outline lifecycle-oriented practices for reproducibility, including containerized environments, declarative workflows, dataset versioning, and model registries with FAIR-compliant metadata. Governance frameworks such as the NIST AI Risk Management Framework and the EU AI Act are discussed as critical scaffolding for risk assessment, transparency, and auditability. By integrating these engineering and scientific perspectives, we propose a structured blueprint for AI-driven modeling stacks that can deliver scalable, verifiable, and regulatory-ready scientific results. This work positions soft matter physics not just as a beneficiary of AI but as a key testbed for shaping robust, reproducible, and accountable computational science.

KEYWORDS

artificial intelligence in science, scientific software engineering, machine learning operations (MLOps), reproducibility and provenance, soft matter modeling, differentiable simulation, physics-informed machine learning, FAIR principles for research software

1 Introduction

Scientific software has long been the connective tissue of modern research. It translates physical hypotheses into executable form, generates predictions from theoretical models, and validates them against experimental or observational evidence. Historically, this role has been anchored by deterministic simulation codes, statistical analysis packages, and workflow managers designed to prioritize correctness, reproducibility, and computational efficiency. Yet in the era of Artificial Intelligence (AI), this picture is being redrawn. At the heart of the change is the growing centrality of data-driven components—statistical models, deep neural networks, and increasingly foundation models—that behave not as fixed algorithms but as mutable dependencies. These models are intimately tied to evolving datasets, training regimes, and hardware environments. Unlike classical scientific codes, which can often remain valid for decades once validated, AI-driven models degrade over time as data distributions shift, dependencies evolve, and compute platforms change. This mutability stresses classical abstractions of scientific software engineering and makes end-to-end reproducibility fragile unless it is carefully designed in from the outset.

The stakes are particularly high in soft matter physics. Soft matter encompasses polymers, colloids, gels, surfactants, liquid crystals, and biomolecular assemblies—systems governed by large thermal fluctuations [1–3], long-lived metastability [4–6], and multiscale dynamics [7–9]. Conventional atomistic simulations struggle to capture their vast spatiotemporal span, often requiring millions of atoms over microseconds or longer. Coarse-graining can extend these scales, but at the expense of introducing model uncertainty and transferability challenges [10–14]. Meanwhile, inverse design problems—such as creating molecules, nanoparticles, or assemblies with targeted functionality—demand exploration of rugged energy landscapes through costly iterative search [15–19]. This confluence of challenges makes soft matter physics a particularly demanding domain for simulation, but also one where AI-enabled approaches promise transformative gains.

In recent years, three converging trends have begun to reshape the practice of soft matter modeling and the software stacks that support it. The first is the ability to learn physical models directly from data. Machine learning architectures such as equivariant graph neural networks and message-passing models can learn force fields and coarse-grained interactions with near *ab initio* accuracy but at the cost of classical molecular dynamics [20–26]. By encoding Euclidean symmetries and, in some cases, long-range physics, these networks respect the fundamental invariances of interatomic interactions [27–30]. Frameworks such as NequIP [31], MACE [32], and Allegro [33] have demonstrated that it is possible to replace hand-crafted potentials with machine-learned surrogates that not only faithfully approximate electronic-structure calculations but also generalize across molecular compositions and thermodynamic states. In the context of soft matter, where transferability is a perpetual challenge, these models open possibilities for constructing coarse-grained descriptions that are physically consistent and data-efficient [34, 35, 82].

The second major trend is the integration of differentiable programming into molecular simulation. Modern toolkits built on JAX or PyTorch, such as JAX-MD [36] and TorchMD [37], expose forces, trajectories, and even loss functions to automatic differentiation [38]. This enables gradient-based optimization across the entire simulation pipeline, turning molecular dynamics into a differentiable layer within larger computational graphs. Such capabilities permit direct parameter learning against experimental observables, efficient sensitivity analysis of dynamic processes, and inverse design workflows in which system parameters are tuned to optimize emergent material properties [38–40]. Differentiable simulation blurs the line between physical modeling and optimization, embedding traditional simulation within the broader ecosystem of differentiable scientific computing.

A third development is the arrival of AI-native software practices into physics laboratories. Research software is increasingly designed around principles drawn from modern machine learning operations, including continuous integration of models and data, automated experiment tracking, and infrastructure for reproducibility. The FAIR for Research Software principles [41] extends the widely adopted FAIR framework—findable, accessible, interoperable, and reusable—from data to code, recognizing that software is not merely auxiliary to science but a primary research object [42, 43]. Alongside this, practices from industry such as Machine Learning Operations (MLOps) [44–46] are being adapted to manage the lifecycle of scientific machine learning models. Versioning, experiment logging, retraining pipelines, and monitoring of model drift are now entering the practice of computational physics [47–49]. Large language models further complicate and enrich this picture, as they increasingly assist with code authoring, testing, and documentation [50–52]. Their integration can enhance productivity but also raises new questions about provenance, correctness, and scientific accountability. Together, these developments accelerate discovery but also foreground challenges of reliability, reproducibility, and validity.

Concerning LLMs, they have begun to play a non-trivial role in scientific code generation, raising a distinct set of challenges that extend beyond mere functionality. For example, provenance questions become acute: when code is scaffolded or entirely generated by an LLM, scholars must ask “Which version of which model contributed?” and “Which training data or prompt context underlies this snippet?” Recent work highlights that standard provenance frameworks often fail to capture agent-centric metadata such as prompts, model version, and contextual lineage [53–55]. Licensing and intellectual-property concerns are also important: LLMs typically draw upon massive corpora of code under heterogeneous licenses, creating ambiguities around downstream reuse, attribution, and derivative work [56–58]. Meanwhile, traceability and version control become more complex when the code evolves through a hybrid human–machine loop: ensuring that subsequent edits, branch merges, and model-assisted contributions remain auditable is non-trivial [53, 59, 60]. Together, these issues emphasize that adopting LLM-generated code in scientific workflows is not just a productivity win, but a shift that demands explicit governance of provenance, licensing, and version-traceability if reproducibility and accountability are to be preserved [55, 61, 62].

These shifts must be understood against the backdrop of a reproducibility crisis [47, 63, 64] that has affected many areas of computational science. The integration of data-driven models introduces new risks that are absent or less pronounced in traditional simulation. Model boundaries [31, 65] are prone to erosion as applications expand beyond the regimes on which training data are drawn. Training and prediction can become entangled with poorly documented datasets, complicating efforts to disentangle provenance and validity [66, 67]. Configurations of hyperparameters, dependencies, and hardware optimizations often accumulate as invisible technical debt, leading to fragility in software reuse [68–70]. Feedback loops can arise when models trained on generated or biased data reinforce their own errors [71–73]. These failure modes are articulated in the influential notion of “hidden technical debt” in machine learning systems, which emphasized the gap between rapid experimental progress and the challenges of long-term maintainability [61, 74]. Unlike classical simulation codes, which once validated can remain useful for decades, AI-driven systems demand constant monitoring, retraining, and governance [68, 70, 75].

In parallel, the reproducibility movement has articulated frameworks to address these risks. The FAIR principles originally applied to data have been extended to research software, emphasizing that both code and models must be findable, accessible, interoperable, and reusable [41, 66]. Communities such as computational biology and software carpentry have emphasized pragmatic rules for sustainable research coding, from documenting provenance to versioning code and data together [42, 48, 49, 76]. These community-driven initiatives demonstrate that reproducibility is not an optional add-on but an essential element of credible scientific practice [63, 77, 78]. Yet AI intensifies the challenge by multiplying the number of mutable dependencies and by entangling software, data, and hardware in ways that make reproducibility nontrivial [47, 67, 79].

Soft matter physics provides a compelling proving ground for tackling these challenges. The field is inherently multiscale, spanning length scales from angstroms to microns and time scales from femtoseconds to seconds [11, 12, 80]. Bridging these scales has always required hybrid methods, combining atomistic, coarse-grained, and continuum representations [80]. AI-enabled approaches promise to unify these levels through learned surrogates and differentiable couplings, offering a route to predictive models that maintain fidelity across scales [32, 81, 82]. The rugged free-energy landscapes [83] that characterize soft matter pose another challenge, as metastability and rare events dominate system behavior [84, 85]. Traditional simulation methods struggle to sample these efficiently [86], but AI-assisted sampling and generative models offer new ways to accelerate exploration. Moreover, many problems in soft matter are inherently inverse: designing polymers for mechanical resilience, nanoparticles for drug delivery, or surfactants for targeted self-assembly all require forward modeling of emergent structures and inverse optimization of component design. These dual demands align closely with the strengths of AI, particularly generative models and gradient-based optimization in high-dimensional spaces. Experimental advances further reinforce this potential, as scattering, microscopy, and single-molecule techniques provide increasingly rich datasets that can inform and validate AI-augmented models [87–96].

Although this work focuses on soft matter systems, Computational Fluid Dynamics (CFD) provides a complementary example of how computational science is being reshaped by AI/ML. Both domains face similar challenges: they require solving nonlinear partial differential equations (Navier–Stokes for fluids, reaction–diffusion or elasticity equations for soft matter) over wide spatiotemporal scales, often under uncertainty. In CFD, machine learning has already shown its potential to accelerate simulations and enable real-time control through physics-informed neural networks (PINNs) [97, 98], neural operators [99], and hybrid turbulence models [100]. These advances demonstrate how embedding physical laws into learning architectures can reduce computational cost, improve generalization, and make design space exploration tractable. The lessons learned from AI-augmented CFD—such as the need for differentiable solvers [101, 102], uncertainty-aware active learning [103], and rigorous reproducibility [104–106]—directly inform the strategies we advocate for soft matter modeling, where similar multiscale and data-efficiency issues arise.

Together, these developments sharpen a central question: *what should good scientific software look like in an AI-intensive world?* The answer cannot simply be to append machine learning modules to existing codes. Rather, what is needed is a lifecycle-oriented approach that integrates domain-specific modeling with robust engineering practices and governance frameworks. Reproducibility must be embedded by design through containerization, workflow automation, and metadata-rich logging. Models must be continuously validated and retrained as data or conditions shift, requiring infrastructure for continuous integration of simulations with evolving datasets. Choices between atomistic, coarse-grained, and AI-augmented models must be guided not only by accuracy requirements but also by sustainability and interpretability. Governance of both data and models must align with emerging standards such as the NIST AI Risk Management Framework [107] and EU Regulation [108], evolving regulatory regimes, ensuring transparency, accountability, and ethical use. Finally, stewardship of scientific software in this new regime must be interdisciplinary, bridging physics, computer science, and data science to ensure both scientific validity and engineering robustness.

The aim of this article is to provide a practitioner-oriented synthesis of these themes. We examine the state of the art in AI-accelerated modeling of soft matter, reviewing developments in machine-learned force fields, differentiable simulation frameworks, and inverse design strategies. We analyze how software engineering practices drawn from MLOps, FAIR4RS (FAIR for Research Software), containerization, and LLM-assisted development can be adapted to scientific contexts. We identify the risks and challenges to reproducibility, maintainability, and governance posed by AI-driven scientific software. From this analysis we propose a set of lifecycle-oriented design principles that integrate modeling needs with reproducibility frameworks and governance standards.

By weaving together these strands, we position soft matter physics not merely as a beneficiary of artificial intelligence but as a proving ground for broader questions of how to build scientific software that is accurate, sustainable, and FAIR in an AI-intensive world.

2 From descriptive to executable knowledge: software as the scientific substrate

Scientific understanding is increasingly embedded in executable artifacts—simulators, learned potentials, analysis notebooks, and automated pipelines—shifting the focus from static publications to dynamic, reproducible workflows [42, 66, 109]. In soft matter, this transition is driven by three core challenges. First, multiscale coupling: soft materials exhibit behavior spanning multiple length and time scales, from molecular interactions to mesoscopic assemblies, necessitating models that propagate information across scales without losing fidelity [80, 110–112]. Second, stochastic dynamics and rare events: phenomena such as nucleation, self-assembly, and folding are dominated by fluctuations, requiring simulation strategies capable of capturing rare but critical events [83, 113, 114]. Third, the richness of design spaces: the combinatorial possibilities of sequences, architectures, and interaction parameters demand tools that can efficiently navigate complex landscapes and identify promising candidates [19, 115, 116].

Modern AI-era software responds to these pressures with a suite of complementary approaches. Composable kernels with automatic differentiation (AD) allow force computations to be fully differentiable, enabling end-to-end optimization against experimental observables or higher-level targets, bridging the gap between local interactions and emergent behaviors [36]. Equivariance by construction, through SO(3) or E (3)-equivariant networks, embeds fundamental rotational and translational symmetries directly into model architectures, enhancing sample efficiency and out-of-distribution robustness, which is critical for predicting novel soft matter configurations [31, 82]. Active learning loops leverage uncertainty-aware sampling to maintain compact yet representative training sets, systematically exploring metastable configurations while capturing rare events efficiently [117–120]. Finally, closed-loop inverse design uses differentiable or surrogate-based objectives to steer simulations toward targeted structures, such as colloidal crystals, block copolymer morphologies, or mesophases, integrating prediction, simulation, and evaluation into a single adaptive workflow [121–123].

In computational fluid dynamics (CFD), analogous challenges arise. Realistic flow problems span wide ranges of length and time scales, often coupling to heat transfer, chemical reactions, or multiphase interfaces. Traditional solvers face steep computational costs due to mesh resolution demands and nonlinearities in the governing Navier–Stokes equations, especially in complex geometries or parameter sweeps. Furthermore, the design spaces in CFD—ranging from aerodynamic shapes to microfluidic architectures—are combinatorially rich, requiring tools that can navigate large parameter landscapes efficiently. Modern AI-driven strategies address these barriers: neural operators [99, 124, 125] and differentiable solvers [126–128] provide fast surrogates for PDE solutions [129], enabling real-time exploration of design spaces while retaining accuracy across varying boundary conditions. Automatic differentiation–enabled CFD kernels [130, 131] allow direct gradient computation through flow simulations, bridging high-fidelity physics with optimization and inverse design objectives. Active learning frameworks [132–134] selectively enrich training sets with high-value simulations, avoiding exhaustive mesh refinements or brute-force parameter sweeps. Finally, closed-loop design and control integrate CFD solvers with differentiable objectives—such as drag reduction, pressure distribution, or flow uniformity—yielding adaptive pipelines where simulation, optimization, and evaluation are tightly coupled [135–137].

Collectively, these advances mark a paradigm shift in scientific research: rather than treating simulation, learning, and design as separate steps, they are increasingly intertwined in adaptive, end-to-end workflows capable of addressing the inherent complexity and stochasticity of physical systems.

3 Examples of AI/ML use in soft matter

3.1 ML interatomic and coarse-grained potentials for soft matter

3.1.1 Equivariant message passing for atomistic resolution

Interatomic potentials are mathematical models that describe how atoms interact—essentially, they provide the energy of a system as a function of atomic positions. Traditional potentials (like Lennard–Jones or EAM) use fixed functional forms. Machine-learning interatomic potentials (MLIPs) replace those with neural networks or Gaussian processes trained on quantum-mechanical data (e.g., DFT). This allows MLIPs to be much more accurate while remaining computationally cheaper than *ab initio* calculations. Equivariant MLIPs have advanced rapidly in recent years. Methods such as NequIP [31], see Figure 1, introduced highly sample-efficient E (3)-equivariant convolutions, while MACE [32] leveraged higher-order message passing to achieve fast, accurate force predictions. Allegro further decoupled network capacity from message passing, scaling to millions of atoms at high throughput [33]. Although initially benchmarked on small-molecule datasets like rMD17, these approaches are increasingly applied to condensed-phase and soft-condensed systems, including those with long-range interactions and charge equilibration effects [138].

For soft matter applications, polymer melts, ionic liquids, and other polar systems require careful treatment of long-range and many-body interactions [138–141]. Modern MLIPs integrate global charge equilibration and cutoff-aware message passing, enhancing transferability across densities and phases. Hybrid invariant/equivariant architectures or explicit polarization models are recommended for hydrated or strongly polar materials [138, 140, 142, 143]. Deployment strategies typically start with compact equivariant architectures (e.g., NequIP-like models) and scale only if model uncertainty remains significant. Active learning, coupled with on-the-fly molecular dynamics, can enrich training sets in regions of high epistemic uncertainty [144–148]. Validation should emphasize kinetic observables, such as transport and relaxation times, rather than force mean absolute error alone, as soft matter is often highly dynamics-sensitive.

3.1.2 Coarse-graining with GNNs

At the mesoscale, coarse-grained (CG) models reduce the number of degrees of freedom while retaining essential thermodynamic and kinetic properties [149]. Graph neural network (GNN)-based CG approaches have emerged [150], enabling simultaneous learning of mapping operators [151–153] and CG force fields [154, 155]. Iterative decoding schemes have also been explored to reconstruct atomistic detail from CG representations, maintaining consistency across resolutions [156, 157]. Recent work by [152], has formalized a workflow for machine learning–based coarse-graining (ML-CG) that integrates data preparation, mapping generation, force-field training, and validation under a unified framework, ensuring systematic benchmarking and thermodynamic transferability across state points. This workflow emphasizes curating representative training data, choosing an expressive yet physically meaningful model class (e.g., message-passing neural networks), and employing rigorous metrics such as free energy differences, structural distributions, and transport coefficients to evaluate CG model quality. A schematic of the ML-CG can be seen in Figure 2a.

In parallel, graph neural network–based mapping prediction has been advanced by frameworks such as the Deep Supervised Graph Partitioning Model (DSGPM) [151], which treats CG mapping as a node-partitioning problem on a molecular graph–see Figure 2b. DSGPM jointly optimizes partition assignments and a supervised loss function to reproduce target CG mappings, incorporating

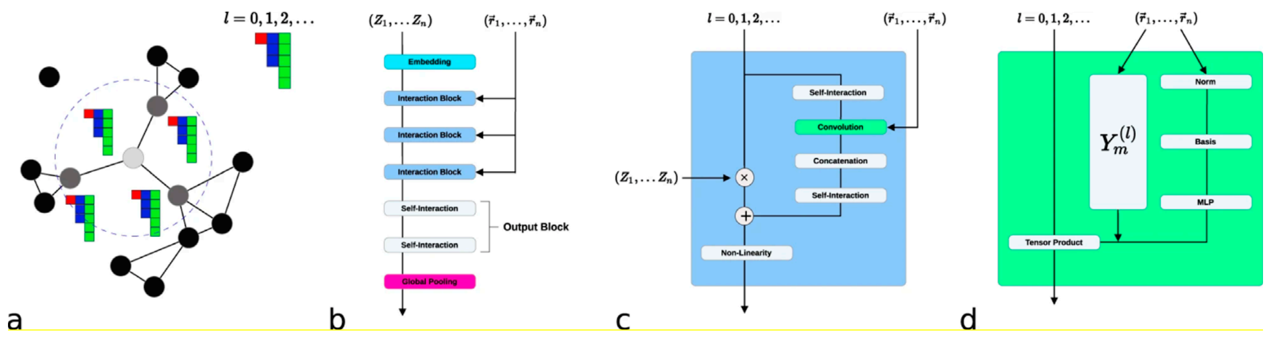

FIGURE 1
The NeqIP network architecture **(a)** A molecular or atomic system is represented as a graph, where nodes correspond to atoms and edges encode pairwise interactions within a specified cutoff, thereby defining local atomic neighborhoods. **(b)** Atomic numbers are embedded into initial scalar features ($\ell = 0$), which are iteratively refined through a sequence of interaction blocks, yielding scalar and higher-order equivariant tensor features. **(c)** Each interaction block applies an equivariant convolution over the local graph to update atomic representations while preserving rotational symmetry. **(d)** The convolution operation is constructed as the tensor product of a learnable radial function *R*(*r*) of the interatomic distance and the spherical harmonic projection of the interatomic unit vector, which is subsequently contracted with neighboring atomic features. The resulting features are processed by an output block to predict atomic energies, which are aggregated to yield the total energy of the system. Reprinted from [31], licensed under CC BY.

chemical priors (e.g., functional groups, ring structures) to produce chemically interpretable partitions. This allows automatic discovery of optimal mapping operators that preserve structural and thermodynamic features while remaining generalizable to new molecules or state points.

Best practices for CG modeling include treating the mapping as a learned graph partition constrained by chemical and physical priors, combining force-matching with relative entropy or thermodynamic constraints to enhance transferability, and employing uncertainty-aware or ensemble training to prevent mode collapse in rugged energy landscapes [81, 158–161].

3.1.3 Differentiable MD engines

Differentiable molecular dynamics engines, such as JAX-MD [36] and TorchMD [37], allow gradients to flow through the simulation loop, enabling direct optimization of parameters against experimental data, trajectory alignment, or inverse design objectives. For soft matter, differentiability facilitates the tuning of coarse-grained potentials, nonbonded interactions, field-theory parameters, and differentiable constraints in mesoscale models, providing a unified framework for learning and simulation [162–165]. Building on this paradigm, differentiable molecular simulation (DMS) approaches–see Figure 3, such as that proposed by Greener and Jones [162], demonstrate that it is possible to learn all parameters in a CG protein force field directly from differentiable simulations. Their work couples a neural-network-based CG potential with a fully differentiable MD integrator, optimizing parameters end-to-end by backpropagating through complete trajectories. This enables direct minimization of loss functions defined on structural and thermodynamic properties (e.g., native-state stability, contact maps) without requiring hand-tuned potentials. The DMS framework thus provides a powerful route to data-driven CG force fields with improved accuracy and transferability, bridging simulation and learning in a single, differentiable pipeline.

3.2 Inverse design and self-assembly

3.2.1 Learning to target phases and structures

Deep learning has become a transformative tool for mapping microscopic interaction parameters or building-block designs directly to emergent structures, enabling a paradigm shift from trial-and-error exploration to data-driven materials discovery. A landmark example is the work by Coli et al. [121], who employed deep-learning-based inverse design to train surrogate models that map colloidal particle interactions to target crystal structures. This approach dramatically accelerates the discovery process compared with brute-force searches in high-dimensional parameter spaces. Their approach can be seen schematically in Figure 4.

Recent follow-up studies extend these methods to increasingly complex systems. For example, Wang et al. [166] applied enhanced sampling techniques in combination with deep learning to predict open and intricate crystal structures that are otherwise difficult to access. Similarly, multiobjective inverse design frameworks now allow simultaneous optimization of competing criteria such as nucleation kinetics, yield, and structural robustness [167], illustrating the growing sophistication of AI-guided materials design.

A common "design loop" pattern has emerged in these workflows:

1. Propose: candidate designs are suggested using surrogates, large language models, or genetic algorithms.
2. Simulate/evaluate: candidates are evaluated through molecular dynamics (MD), Monte Carlo (MC), density functional theory (DFT), or self-consistent field theory (SCFT) simulations.
3. Update: the design model is refined using Bayesian optimization or active learning strategies to prioritize promising regions of parameter space.
4. Iterate: the loop continues until design constraints or target properties are satisfied.

Increasingly, differentiable simulation engines enable gradients of observables—such as order parameters or structure factors—to be

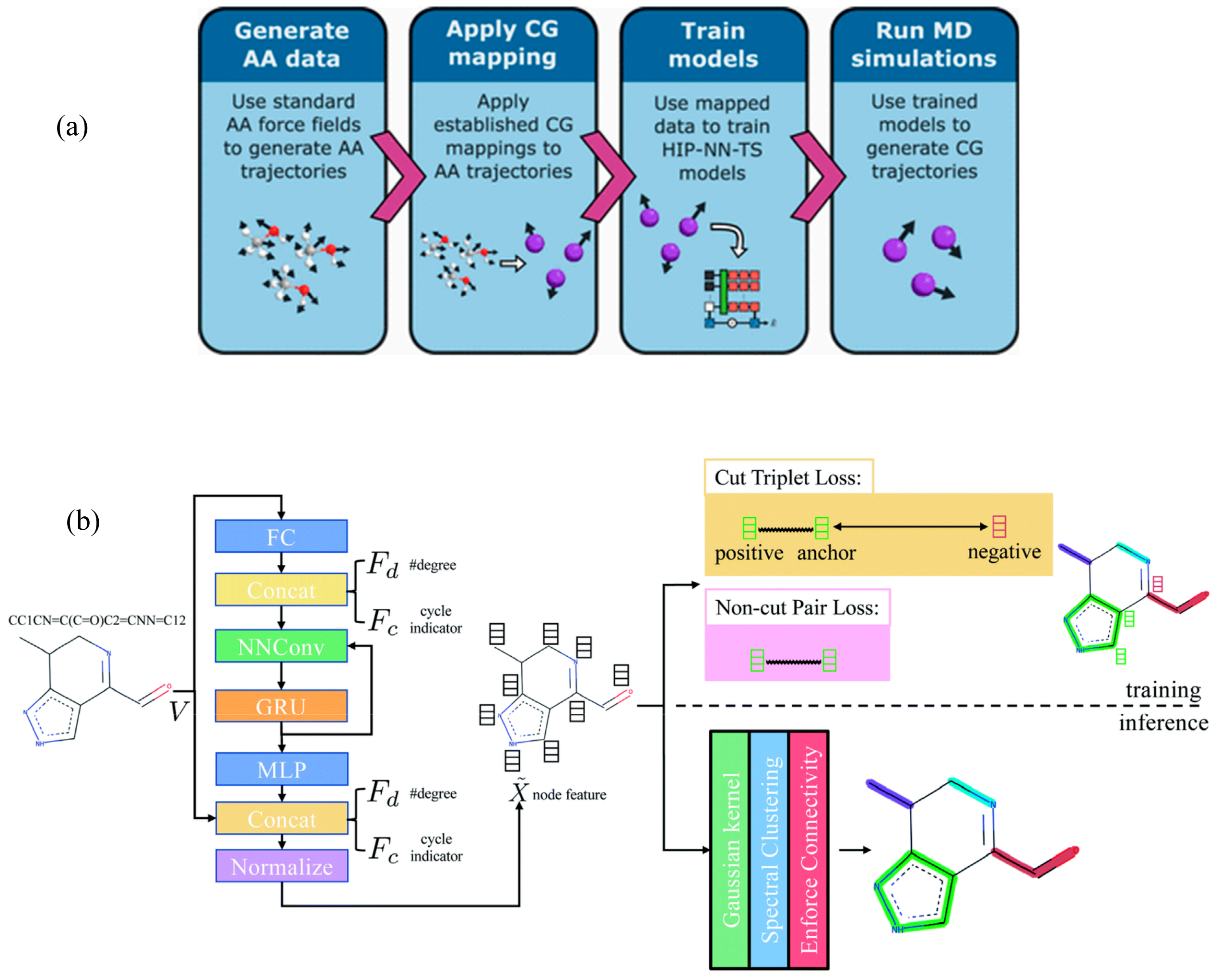

FIGURE 2
(a) Workflow for machine learning–based coarse-graining (ML CG). Atomistic (AA) molecular dynamics simulations are first performed to generate trajectory data, which are mapped to a coarse-grained (CG) representation via a chosen mapping scheme (here, center-of-mass mapping). Reference forces on the CG beads are then computed using the force-matching (multiscale coarse-graining) formalism. These force–configuration pairs form the training data for a machine learning model based on the Hierarchically Interacting Particle Neural Network with Tensor Sensitivity (HIP-NN-TS) architecture, which learns the many-body CG force field. The trained model is then used to run CG molecular dynamics simulations, and the resulting structural and thermodynamic properties (e.g., radial distribution functions) are compared against AA results to evaluate model accuracy and transferability across state points. Reprinted from [152], licensed under CC-BY-NC-ND 4.0. **(b)** Overview of the Deep Supervised Graph Partitioning Model (DSGPM) framework for coarse-grained mapping prediction. The molecular graph is first encoded by projecting one-hot atom and bond types into an embedding space and augmenting with atom degree and cycle indicators. These features are iteratively updated through message-passing neural network layers (NNConv) and gated recurrent units (GRU) to produce atom embeddings. The embeddings are L2-normalized and combined to form a learned affinity matrix, which is then used in a spectral clustering step to predict coarse-grained partitions. Training is guided by a metric-learning objective combining a cut triplet loss (encouraging separation of atoms across CG beads) and a non-cut pair loss (encouraging similarity of atoms within the same CG bead). Reprinted from [151], licensed under CC BY 3.0.

backpropagated directly to design parameters, further accelerating convergence and allowing for end-to-end differentiable inverse design pipelines.

3.2.2 DNA-programmed and patchy colloids

Programmable DNA linkers and patchy colloids offer unprecedented control over interparticle interactions, making them a natural playground for AI-assisted self-assembly design. By encoding specific binding patterns into DNA strands or particle patches, researchers can direct the formation of complex structures with nanoscale precision. Recent demonstrations include the seeded growth of DNA-coated colloids into macroscopic photonic crystals [168], highlighting the potential for optical materials with tunable properties. In parallel, Liu et al. [169] demonstrated the inverse design of DNA-origami lattices using computational pipelines, where ML-guided design predicts sequences and assembly pathways that yield target lattice architectures. These approaches exemplify how the combination of programmable building blocks and AI enables the rational engineering of functional materials from the bottom up.

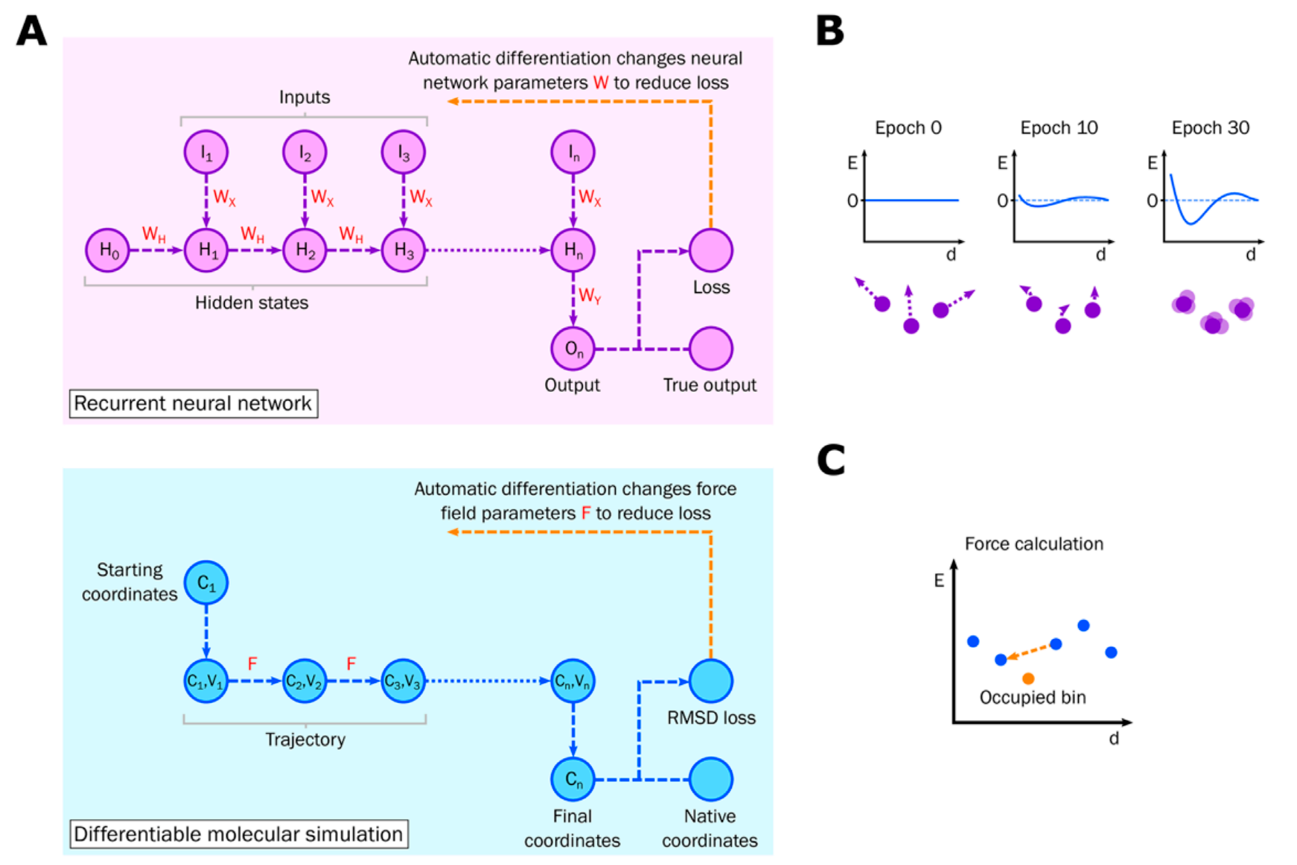

FIGURE 3
Differentiable molecular simulation (DMS) method. **(A)** Analogy between a recurrent neural network (RNN) and DMS, where the same learnable parameters are reused at each step. **(B)** Learning the potential: a schematic shows how the energy function, initially flat at the start of training, adapts during training to stabilise native protein structures. **(C)** Force calculation: forces are derived from the learned potential using finite differences between adjacent energy bins, driving atoms toward native-like distances. Reprinted from [162], licensed under CC BY 4.0.

3.2.3 Polymer sequence and architecture design

In soft materials, polymer sequence and architecture strongly influence macroscopic properties, from mechanical performance to self-assembled morphologies. Active-learning workflows now explore vast polymer sequence spaces by coupling physics-based simulators with ML surrogates, forming closed-loop systems where computational predictions inform experimental synthesis and *vice versa* [170]. In parallel, molecular simulation of coarse-grained systems using machine learning has emerged as a powerful strategy to accelerate exploration of chemical and conformational space. By leveraging ML-derived force fields or coarse-grained potentials, these methods can capture essential molecular interactions at a fraction of the computational cost of all-atom simulations. When integrated into active-learning pipelines, such coarse-grained ML models enable rapid screening of polymer sequence variants, identification of design rules, and prediction of emergent material properties under experimentally relevant conditions [159].

Coarse-grained and reverse-mapped simulations of long-chain atactic polystyrene [171] show how coarse models parameterized via iterative Boltzmann inversion can replicate chain conformation, thermodynamics, and entanglements in high molecular weight melts. Work on isotactic polypropylene [172] shows how coarse grain potentials can be crafted to preserve fine structural features like helicity. On the dynamics/rheology side, the viscoelastic behavior of melts has been studied in multiscale frameworks (e.g., expanded ensemble Monte Carlo + nonequilibrium MD) [173] to access mechanical and time-dependent properties. A recent ML-hybrid coarse-graining framework [174] directly ties into these strategies, enforcing both bottom-up structure matching and top-down density/thermodynamics, while delivering transferability.

These collective efforts illustrate how ML methods are increasingly able to augment and extend older hierarchical/coarse-grained simulation approaches. By combining efficient sampling (connectivity altering MC, reverse mapping), richly parameterized CG force fields, and ML surrogates/optimization loops, one can push toward materials design frameworks that not only predict but *optimize* polymer sequence and architecture for target behaviors (mechanical moduli, morphology, rheology, etc.), possibly with experimental feedback in closed form.

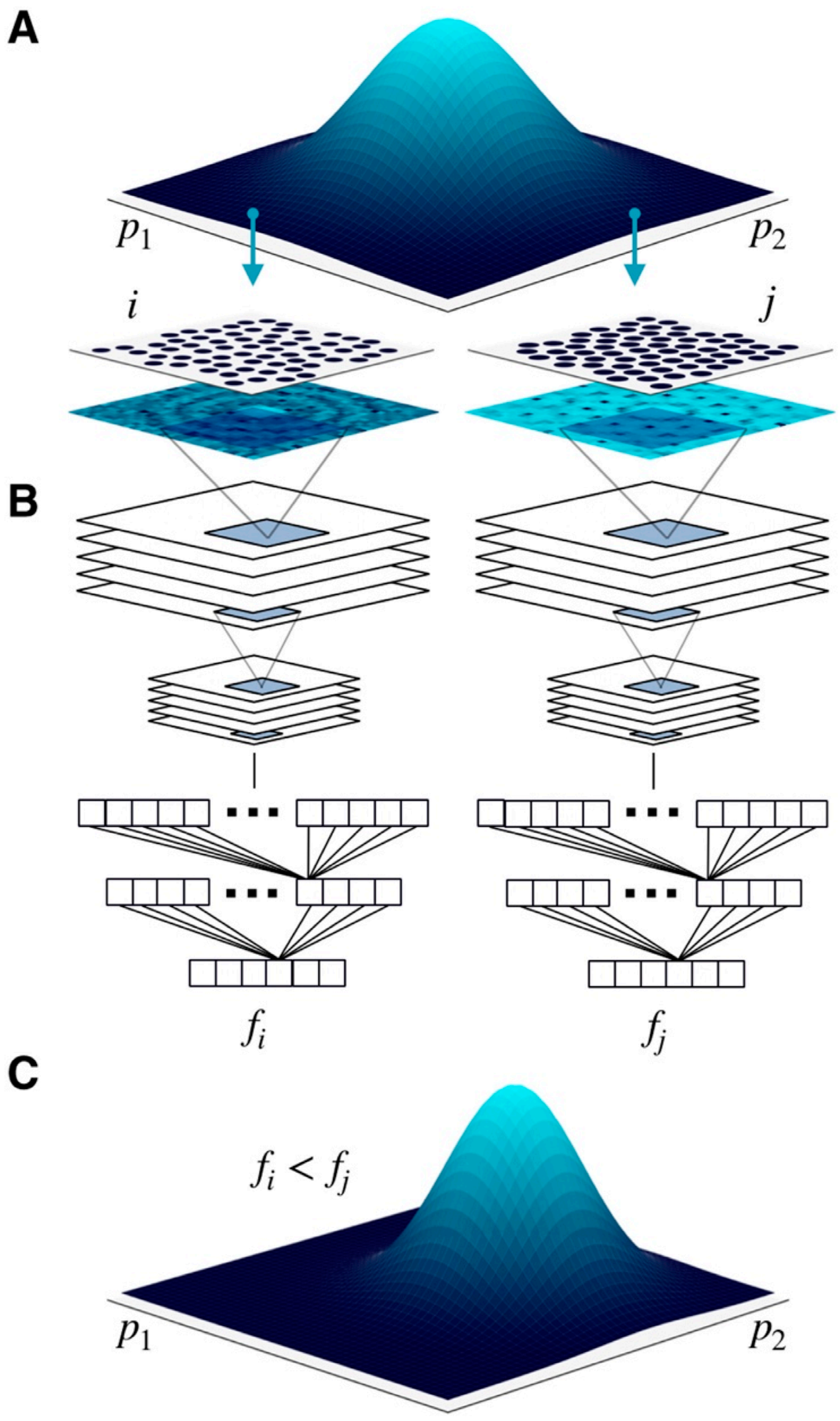

FIGURE 4
(A) Candidate interaction parameters are drawn from a multivariate Gaussian distribution, and each set is used to run a simulation of particle self-assembly. **(B)** Configurations from these simulations are evaluated by a convolutional neural network (CNN) trained to classify phases from their diffraction patterns, providing a fitness score based on similarity to the target structure. **(C)** The covariance matrix adaptation evolutionary strategy (CMA-ES) updates the Gaussian distribution, shifting it toward regions of parameter space that yield higher fitness, thereby iteratively refining the design until the target phase is stabilized. Reprinted from [121], licensed under CC BY 4.0.

3.3 Physics-informed machine learning and hybrid models in computational fluid dynamics

The rapid progress of ML/AI has opened new opportunities in computational fluid dynamics (CFD), where traditional numerical solvers often face challenges related to computational cost, high-dimensional parameter spaces, and complex nonlinear dynamics. Physics-informed machine learning (PIML) offers a promising path forward by embedding governing physical laws, such as the Navier–Stokes equations, directly into learning architectures. This integration enables models to generalize better, respect conservation principles, and require less data than purely data-driven approaches. Hybrid models, which combine conventional numerical solvers with ML components, further enhance predictive accuracy and efficiency, making them attractive tools for turbulence modeling, multiscale simulations, and real-time flow prediction.

3.3.1 Physics-informed machine learning in fluid mechanics

Physics-Informed Machine Learning (PIML) represents a paradigm in which governing equations, symmetries, and physical constraints are embedded directly into the learning process. By integrating physics into the model architecture or loss functions, PIML approaches reduce the reliance on large datasets and mitigate unphysical predictions in extrapolative regimes. The article by Karniadakis et al. [98] played a pivotal role in catalyzing interest in these methods, providing a comprehensive overview of approaches such as physics-informed neural networks (PINNs), operator learning, and symbolic regression for equation discovery [98, 175–177].

Applications of PIML are rapidly expanding into soft condensed matter systems, where accurate modeling of complex fluids and materials is often limited by experimental sparsity or computational cost. Operator learning and neural operators can efficiently approximate field-to-field mappings, such as predicting stress or flux distributions from concentration or velocity fields, across varying geometries and boundary conditions [178–182]. In parallel, hyperdensity functional theory (HDFT), which generalizes classical density functional theory to arbitrary thermal observables, has been combined with ML representations to accelerate the computation of thermodynamic properties and phase behavior [183–186].

PINNs–see Figure 5–and hybrid residual models have shown particular promise in this domain. By enforcing the underlying differential equations as constraints, PINNs can regularize sparse experimental or simulation data, providing physically consistent interpolations [175, 187, 188]. Hybrid residual models, where machine learning predicts corrections to a baseline theory such as DFT or self-consistent field theory (SCFT), allow the combination of robust theoretical frameworks with data-driven flexibility, enhancing generalization in low-data regimes [189–194].

For soft matter applications, it is particularly important to encode symmetries and conservation laws early in the modeling process, as this reduces the risk of unphysical extrapolations. Operator learning is advantageous for field-to-field mappings, since neural operators can efficiently generalize across different geometries and boundary conditions [178, 179, 196]. At the same time, hybrid residual models that learn corrections to existing theoretical frameworks improve predictive accuracy while maintaining consistency with established physics [189, 190, 192]. Emerging trends in PIML for soft matter include the integration of machine learning with coarse-grained simulations and kinetic models, which enables the study of multiscale phenomena and complex non-equilibrium processes with greater efficiency and accuracy [180, 181, 194].

3.3.2 Integration of ML in CFD

In the realm of CFD, the integration of machine learning techniques has been transformative. Machine learning models, particularly PINNs, have been employed to solve complex fluid dynamics problems governed by partial differential equations (PDEs) [98]. These models incorporate physical laws directly into the learning process, allowing for accurate predictions even with limited data.

For instance, in the study by Cai et al. [180], PINNs are applied to simulate three-dimensional wake flows and supersonic flows, demonstrating their capability to handle complex fluid dynamics scenarios. Similarly, in biomedical applications, PINNs have been utilized to model blood flow in arteries, providing insights into hemodynamics without the need for extensive experimental data.

Moreover, recent advancements have focused on enhancing the training strategies for PINNs to improve their performance in stiff fluid problems. A novel approach called "re-initialization" has been proposed by Raisi et al. [175], which periodically modulates the training parameters of the PINN model. This strategy enables the model to escape local minima and effectively explore alternative solutions, leading to more accurate and physically plausible results in simulations of fluid flows at high Reynolds numbers.

3.3.3 Hybrid models and surrogates in CFD

Hybrid models that combine traditional CFD methods with machine learning techniques have shown promise in accelerating simulations and improving accuracy. These models leverage the strengths of both approaches: the physical rigor of CFD and the data-driven adaptability of machine learning.

For example, in turbulence modeling, machine learning algorithms such as Tensor Basis Random Forests (TBRF) have been used to predict Reynolds stress anisotropy and integrated into Reynolds-Averaged Navier–Stokes (RANS) solvers, yielding results close to Direct Numerical Simulation/Large Eddy Simulation (DNS/LES) references with physical consistency [197]. Physics-informed ML approaches have also been developed to correct discrepancies in RANS-modeled Reynolds stresses using DNS data, improving predictions in high-Mach-number turbulent flows [198]. Iterative ML-RANS frameworks that seamlessly integrate ML algorithms with transport equations of conventional turbulence models have demonstrated enhanced predictive capability across varying Reynolds numbers, including separated flows [199]. Broader reviews underscore the blend of ML corrections and classical closure modeling—for example, through tensor basis neural nets, gene-expression programming, and physically informed field inversion—highlighting the importance of preserving physical compatibility while enhancing model flexibility [200].

Additionally, machine learning-based surrogates have been developed to replace expensive CFD simulations in optimization problems. These surrogate models rapidly approximate flow

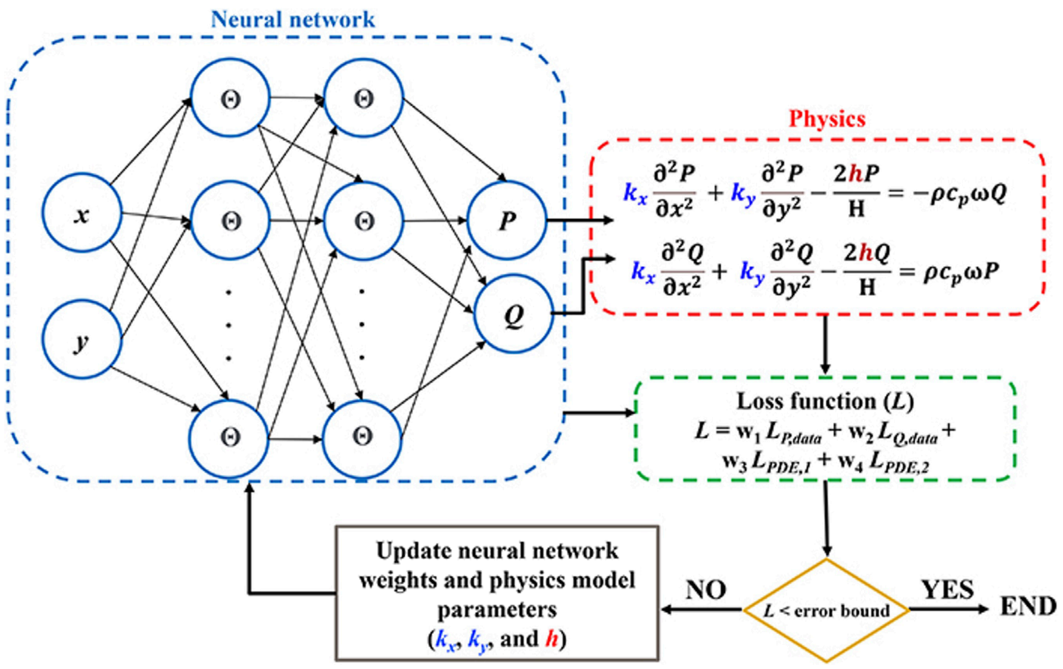
Neural network
x
y
Θ
P
Q
Physics
$k_x \frac{\partial^2 P}{\partial x^2} + k_y \frac{\partial^2 P}{\partial y^2} - \frac{2hP}{H} = -\rho c_p \omega Q$
$k_x \frac{\partial^2 Q}{\partial x^2} + k_y \frac{\partial^2 Q}{\partial y^2} - \frac{2hQ}{H} = \rho c_p \omega P$
Loss function (L)
$L = w_1 L_{P,data} + w_2 L_{Q,data} + w_3 L_{PDE,1} + w_4 L_{PDE,2}$
Update neural network weights and physics model parameters (k_x, k_y, and h)
NO
L < error bound
YES
END

FIGURE 5
Physics-Informed Neural Networks (PINNs) are a class of machine learning models that incorporate physical laws, typically expressed as partial differential equations (PDEs), into the training process of neural networks. Unlike traditional neural networks, which rely solely on data-driven learning, PINNs embed the governing equations of the system directly into their loss function. This allows them to simultaneously fit available observational data and enforce consistency with the underlying physics. In practice, the input to a PINN (e.g., spatial or temporal coordinates) passes through a neural network to predict a physical quantity of interest (such as velocity, pressure, or displacement). The model then computes derivatives of this output with respect to the inputs using automatic differentiation. These derivatives are used to evaluate the residuals of the governing PDEs, which are minimized alongside the data mismatch during training. By combining data with physical constraints, PINNs can achieve robust generalization, require less training data, and provide physically consistent solutions even in regions where data is sparse or unavailable. This makes them particularly powerful for scientific computing, engineering design, and modeling complex systems governed by physics. In the figure, input spatial coordinates x and y are fed into the neural network, which produces corresponding outputs of the real and imaginary parts of the complex physical quantities (P and Q). During each iteration, the neural network computes the derivatives using automatic differentiation, while also incorporating physics-based regularization. After each iteration, the loss function (L) is updated to include weighted losses from both the neural network and the physics PDE residuals. Training of the neural network continues until the loss function falls below a specified tolerance level (error bound) and stops once the tolerance is achieved. Reprinted from [195], licensed under CC BY.

behaviors across design parameters, enabling efficient design iteration in engineering contexts [201]. Comparative studies indicate that while polynomial regression surrogates are efficient and suitable for capturing interaction effects, Kriging-based models often offer superior exploration and predictive accuracy across complex design spaces [202].

4 Why AI changes scientific software?

Scientific software has always mediated between hypotheses and evidence. What is different now is the centrality of data-driven components—statistical models, deep nets, and increasingly, foundation models—that behave like *mutable dependencies* tied to evolving data and compute environments. This mutability stresses classical software engineering abstractions and makes end-to-end reproducibility harder if not explicitly designed in from the start. The work by Sculley et al. [68] on “hidden technical debt” in ML systems captures the mismatch between quick experimental wins and long-term maintainability, highlighting ML-specific risks such as boundary erosion, data entanglement, configuration debt, and feedback loops.

In parallel, the reproducibility movement codified the FAIR principles for research objects, while the computational biology and software carpentry communities articulated pragmatic rules for reproducible analysis and “good enough” practices for everyday research code [61, 203].

When LLMs are leveraged for code generation in scientific workflows, additional layers of complexity emerge around verification protocols, documentation standards, and bias mitigation. First, ensuring *physical correctness* of LLM-generated code (for example, that a simulation correctly enforces conservation laws or boundary conditions) may require formal verification or embedded test-suites. Councilman et al. [204] proposes formal query languages and symbolic interpreters to verify that generated code matches user intent. Second, documentation standards must evolve: contributions from generative AI should be explicitly versioned, timestamped, and tagged with model-identifier and prompt context, so that audits can trace “human vs. model vs. hybrid” authorship. Provenance frameworks like PROV-AGENT [53] underscore the need to capture prompts, agent decisions and model versions in end-to-end workflows. Third, scientific workflows must account for *model bias and automation-bias*: for instance, LLMs may over-produce boilerplate or mirror biases embedded in their training code corpora (e.g., skewed co-authorship networks, under-representation of certain methodologies). Auditing and bias-mitigation frameworks for LLMs emphasise transparent documentation of training data sources, limitations, and periodic external review [205]. In sum, embedding LLMs into scientific computational pipelines must be accompanied by codified verification protocols, upgraded documentation practices, and bias-aware governance if we are to maintain scientific integrity, reproducibility, and accountability.

These trends converge on a question: *what should "good" scientific software look like in an AI- intensive world?* This paper proposes a lifecycle-oriented answer grounded in peer-reviewed methods, industry-tested practices, and governance frameworks.

4.1 Research agenda and open issues

Scientific software increasingly incorporates ML/AI components, introducing unique challenges that traditional software engineering practices do not fully address. These challenges arise from the centrality of data, non-deterministic computation, complex provenance requirements, and evolving governance expectations. Data must be treated as a first-class dependency, as ML/AI systems are exceptionally sensitive to the distributional properties of their training and evaluation datasets. As datasets evolve—through the addition of new samples, schema revisions, error corrections, or re-labeling—model behavior may shift in ways that compromise replicability. Without explicit dataset versioning and documentation, reproducing earlier results becomes infeasible. Sculley et al. highlighted data dependencies as a key source of "hidden technical debt" in ML systems, and recent work on data versioning emphasizes that it is not enough to track file revisions: schema, feature encodings, and labeling functions must also be recorded to preserve the semantics of the experiment [206].

ML/AI pipelines further complicate reproducibility because of their inherent non-determinism and hardware variability. Stochastic elements such as random initialization, minibatch shuffling, parallel execution order, and non-deterministic GPU kernels can lead to divergent model outcomes. Moreover, floating-point arithmetic differences across hardware accelerators exacerbate this variability. Exact bitwise reproducibility may therefore be infeasible, but statistical reproducibility—achieved through reporting distributions, confidence intervals, and variance across multiple runs—remains attainable. The reproducibility literature recommends controlling random seeds, pinning library versions, and specifying hardware to mitigate uncontrolled variability [66, 207].

Ensuring reproducibility also requires careful experiment tracking and provenance capture. Reproducible computational science depends on documenting the full set of experimental degrees of freedom: hyperparameters, preprocessing pipelines, data versions, model checkpoints, software stack, and hardware configuration. Provenance-aware workflows, supported by tools such as DVC or Collective Knowledge, enable rigorous auditability and repeatability [208]. Neglecting this leads to "configuration debt," where small, undocumented changes produce irreproducible results [68].

Scientific credibility also depends on robust benchmarking and statistical rigor. Over-tuning to a single benchmark, omitting variance estimates, or reporting only the best-case result risks producing fragile conclusions. Best practice calls for repeated trials, ablation studies, and reporting of uncertainty intervals to distinguish genuine improvements from noise [67, 209].

Moreover, when ML/AI systems are applied to sensitive domains such as health, climate modeling, or biosecurity, they carry significant societal risks. The NIST AI Risk Management Framework (AI RMF 1.0) provides a voluntary approach for documenting and mitigating risks [107], while the EU AI Act establishes a binding risk-tiered regulatory regime with explicit obligations for high-risk system [108]. Both frameworks emphasize documentation, monitoring, incident response, and human oversight as first-class concerns for scientific ML software.

Finally, long-term maintainability remains a pressing concern, as ML pipelines accumulate technical debt over time—often more quickly than traditional scientific code due to data entanglement, rapidly evolving dependencies, and *ad hoc* experimentation. Empirical studies of self-admitted technical debt in ML software show that data preprocessing and model training code are common sources of persistent debt that hinders maintainability and reproducibility [59]. Designing modular, testable components and adopting continuous integration practices are therefore critical for sustainable scientific AI development.

4.1.1 Standardized evaluation sets and living benchmarks

Scientific AI would benefit from evaluation suites that are durable, community-maintained, and transparent about their evolution. Benchmarks should include versioning, change logs, and clear governance for updates so that results remain comparable over time. For example, the work by Olson et al. [210] is a curated, evolving collection of datasets that standardizes dataset formats and provides meta-feature analyses to expose gaps in benchmark diversity. Similarly, Takamoto et al. [211] introduces a suite of tasks based on partial differential equations; it provides both simulation data and baselines, along with extensible APIs so future contributions and new PDE tasks can be added. Another recent work by Wang et al. [212], argues for carefully curated benchmark suites (rather than single-score leaderboards) to expose trade-offs in fairness metrics and prevent superficial optimization. Yet in many scientific fields, such living benchmarks are still rare; community stewardship, funding, and infrastructure to host evolving benchmark standards remain gaps.

4.1.2 Auditable training data and provenance

While standardized benchmarks help evaluate models fairly, transparency about training data provenance is equally critical. Transparency in the provenance of training data—including licensing, consent, content authenticity, and lineage—has become increasingly recognized as critical, especially as models consume large, diverse, and partially opaque datasets. A number of recent works provide promising solutions, but implementation remains sporadic and many gaps persist. One such system is Atlas [213], which proposes a framework enabling fully attestable ML pipelines. Atlas leverages open specifications for data and software supply chain provenance to collect verifiable records of model artifact authenticity and end-to-end lineage metadata. It integrates trusted-hardware and transparency logs to enhance metadata integrity while preserving data confidentiality during pipeline operations. Another is PROV-IO+ [214], which targets scientific workflows on High Performance Computing (HPC) platforms. PROV-IO + offers an I/O-centric provenance model covering both containerized and non-containerized workflows, capturing fine-grained environmental and execution metadata with relatively low overhead. Automated provenance tracking in data science scripts is also addressed by tools like Vamsa [215], which extract provenance information from Python scripts without requiring changes to user code; for example, Vamsa infers which dataset columns were used in feature/label derivations with high precision and recall. A more domain-spanning review is by Gierend et al. [216], which examines the state of provenance tracking in biomedical contexts. They find that although many frameworks and theoretical models exist, completeness of provenance coverage is often lacking—especially concerning licensing, consent, and downstream lineage metadata. Another systematic literature review that focuses specifically on healthcare systems is by Ahmed et al. [217]. It highlights how GDPR and similar regulatory pressures increase demand for provenance, but notes that many current provenance technologies fail to capture sufficient context (who created data, under what consent, under what license) or do so in ways that do not scale. Finally, Schelter et al. [218] propose correctness checks and metadata computation based on the provenance of training data and feature matrices; for instance, determining whether test and training sets originate from overlapping data records via lineage queries, which helps guard against data leakage.

4.1.3 Debt-aware design

Machine learning systems accumulate technical debt in many forms: tangled pipelines, configuration debt, insufficient tests, and feedback loops that degrade model quality over time. Empirical studies are starting to quantify this. For instance, Bhatia et al. [59] show that ML projects often admit twice as much debt (in code comments) as non-ML projects, particularly in data preprocessing and model generation components. Another recent work by Pepe et al. [219] categorizes DL-specific technical debt, showing patterns of suboptimal choices tied to configuration, library dependencies, and model experimentation. Further, Ximenes et al. [220] identifies dozens of specific issues in ML workflows that contribute to debt, with data preprocessing being especially debt-heavy. The community would benefit from standardized metrics for ML debt, tool support for detecting and refactoring debt, and best practices to avoid it.

4.1.4 Executable publication

End-to-end re-executability is essential for reproducibility in scientific AI. Interactive formats like Jupyter notebooks improve transparency but often suffer from hidden state, missing dependency specifications, and *ad hoc* execution order, which introduce drift unless workflows are version-pinned and validated via continuous integration (CI). Wang et al. [221] report that over 90% of notebooks lack declared dependency versions (SnifferDog can recover many missing environment metadata); Saeed Siddik et al. [222] document hidden state and out-of-order execution and recommend explicit environment files plus CI; and Grayson et al. [223] quantify that a significant fraction of workflows from curated registries fail due to missing dependencies, environment mismatches, or undocumented configuration—even when containerized.

Publication models increasingly treat code, data, and workflows as a single research object: Peer et al. [224] propose integrating artifact review and metadata creation into publication, but adoption is uneven due to limited infrastructure and weak journal policies. The FORCE11 Software Citation Principles [203] define minimal metadata for citable software (identifiers, versioning, authorship, access, persistence), and journals such as *SoftwareX* and *JOSS* now support software publications. Nonetheless, reproducibility studies [221, 223] show many notebooks still fail to pin dependencies or reproduce reliably, underscoring the need for explicit dependency management and CI-based execution to ensure long-term recomputability.

4.1.5 Regulatory-grade documentation

As regulatory regimes such as the EU AI Act come into force, the need for documentation that meets audit and compliance requirements—beyond good research reporting—has become increasingly clear. Existing practices like Model Cards and Datasheets provide useful scaffolds but often lack the completeness or structure demanded by regulation. Recent work seeks to make documentation "regulatory-gr ade" by design. Lucaj et al. [225] introduce open-source templates for documenting data, model, and application components across the AI lifecycle, explicitly aligning with EU AI Act technical documentation requirements to ensure traceability and reproducibility. Golpayegani et al. [226] propose a framework for human- and machine-readable risk documentation that is interoperable, updatable, and exchangeable between stakeholders. Brajovic et al. [227] extend Model and Data Cards with "use-case" and "operation" cards to better support certification, third-party audits, and compliance. Bogucka et al. [228] contribute an impact assessment template grounded in the EU AI Act, NIST AI RMF, and ISO 42001, co-designed with practitioners and shown to be more comprehensive than baseline reports.

AI governance in scientific contexts benefits from formal frameworks that ensure ethical, robust, and legally compliant systems. The NIST AI Risk Management Framework (AI RMF 1.0) [107] provides a voluntary but comprehensive structure around governance, risk identification, measurement, and monitoring. Scientific teams can adapt it by maintaining risk registers for failure modes (e.g., hallucination, prompt injection, content provenance errors), defining internal controls, and assigning governance responsibilities. Its Generative AI profile highlights risks specific to foundation models, such as misleading outputs and adversarial attacks, and suggests mitigation strategies.

The European Union's AI Act (Regulation (EU) 2024/1689) establishes a binding, tiered risk regime. High-risk AI systems must provide technical documentation, implement rigorous data governance (with representative and accurate datasets), enable human oversight, ensure robustness, and perform post-market monitoring. The Act came into force in August 2024, with most high-risk obligations phased in by August 2026 [229]. Scientific software in high-risk domains—such as medical devices or decision-support systems—must align development workflows with these requirements, including dataset annotation, evaluation documentation, deployment traceability (e.g., logs), and incident monitoring. These regulations do not replace sound engineering practice but formalize it, embedding documentation, monitoring, and traceability as core, non-optional elements of trustworthy scientific AI development.

4.2 From reproducibility to recomputability: principles that travel

Reproducible computational science requires that code, data, parameters, and environments be published in a form that others can rerun. Seminal commentaries and "Ten Simple Rules" have made this agenda mainstream across fields. In practice, recomputability relies on four pillars: (i) capture of code + environment; (ii) declarative workflows; (iii) verifiable provenance; and (iv) complete reporting (92,93).

4.2.1 Environments and packaging

Managing computational environments and dependencies is a persistent challenge in reproducible research. Tools like ReproZip [230] help alleviate this by tracing operating system calls to automatically capture all the files, libraries, environment variables, and configuration parameters used during an experiment, then packaging them into a portable bundle [231]. This portable bundle can be unpacked and executed on a different system—be it via Docker, Vagrant, or chroot setups—allowing reviewers or other researchers to reproduce results without manually installing dependencies [232]. Importantly, these generated packages include rich metadata and a workflow specification, which facilitate not only reproducibility but also review by enabling reviewers to explore and vary experiments with minimal effort [233]. In addition, extensions allow bundles to be unpacked and executed through a web browser interface, further lowering the barrier for reproducibility and streamlining reproducibility checks during journal review processes [232].

4.2.2 Executable narratives

Tools like Jupyter notebooks have become one of the most influential tools for open and reproducible science, popularizing the concept of executable papers [234] — documents that interleave prose, code, and results in a single narrative. This format promotes transparency by allowing readers to inspect methods, rerun analyses, and explore alternative scenarios interactively. Their adoption spans a wide range of domains, from bioinformatics and computational physics to social sciences and education [49, 235–237]. However, the growing reliance on notebooks has also triggered systematic studies of reproducibility, revealing common pitfalls such as hidden state, missing dependencies, or non-deterministic outputs [238]. To address these issues, best practices have been proposed, including pinning dependencies to specific versions to avoid "code rot," parameterizing notebooks to make analyses reusable across datasets and experiments, and enforcing continuous execution (e.g., restarting kernels and running all cells in order before publishing) to ensure a clean, deterministic workflow [237, 239]. Collectively, these practices are helping to transform notebooks from *ad hoc* exploratory artifacts into rigorously reproducible and reviewable scientific records.

4.2.3 Workflows

Scientific workflow management systems such as Snakemake [240] and Nextflow [241] enable users to define pipelines in a declarative manner, transforming a collection of individual computational tasks into coherent, reproducible workflows. In these frameworks, each step—or *rule* in Snakemake and *process* in Nextflow—is declared in terms of its inputs, outputs, and execution logic, while the workflow engine orchestrates the execution order and handles dependencies automatically [242]. This declarative abstraction brings clarity and maintainability to pipeline design, facilitates reproducibility through versioning and container integration, and enables seamless scalability across local, HPC, and cloud environments [243], aligning naturally with FAIR and provenance capture [61]. Furthermore, the use of domain-specific languages built on familiar programming languages—Python for Snakemake and Groovy for Nextflow—strikes a balance between expressivity and readability, appealing to bioinformatics practitioners with programming proficiency.

4.2.4 Reporting standards

FAIR, Model Cards, and Datasheets. Reporting standards are a cornerstone of transparent and trustworthy AI and data-driven research. The FAIR principles provide a widely adopted "north star" for managing data, code, and metadata, emphasizing persistent identifiers, rich machine-readable metadata, provenance tracking, and use of shared vocabularies [63, 66, 203, 244]. While FAIR primarily addresses the technical infrastructure required to make research objects reusable, additional community-driven efforts have emerged to capture the social and contextual dimensions of machine learning artifacts. Among these, Model Cards provide structured documentation for trained machine learning models, summarizing intended uses, performance metrics, trade-offs, and known limitations to guide responsible deployment [245]. Similarly, Datasheets for Datasets standardize dataset reporting by describing motivation, composition, collection methods, preprocessing steps, and potential biases, enabling users to assess suitability and ethical implications before use [246, 247]. Together, these frameworks complement FAIR by adding human-interpretable context and domain-specific guidance. Recent extensions have sought to broaden this ecosystem. Proposals like "FAIR 2.0" emphasize semantic interoperability and machine-actionable services that enhance cross-domain data integration [244], while specialized frameworks such as DAIMS [248] focus on reproducibility and regulatory compliance in medical and clinical AI datasets. Collectively, FAIR, Model Cards, Datasheets, and their extensions are converging toward a multi-layered approach to reporting, where technical, ethical, and contextual considerations are captured in complementary ways to support reproducibility, interpretability, and accountability in computational research.

4.3 Architecture patterns for AI-enabled scientific software

4.3.1 Data-centric pipelines

Modern scientific ML systems increasingly treat data as a first-class asset and design pipelines around its controlled evolution. Raw experimental or observational data are stored in read-only "immutable zones," protected by cryptographic digests to ensure integrity. From these immutable sources, derived datasets are curated, schema-validated, and versioned, with their lineage, consent information, and known hazards documented in datasheets [246]. The orchestration of data extraction, transformation, training, and evaluation steps is typically expressed as a declarative directed acyclic graph (DAG) using workflow engines such as Snakemake or Nextflow. Each pipeline stage is coupled with explicit environment specifications—using Conda environments or container images—which strengthens portability and reproducibility across systems [249]. To guarantee consistent experimental outcomes, dataset and feature versioning are adopted as standard practice. Content-addressed storage and snapshotting ensure that identical data hashes yield identical training sets, eliminating ambiguity introduced by evolving datasets. Feature stores with explicit contracts enforce stable, well-documented feature definitions and prevent silent breakages when features change. Each trained model artifact is coupled with a manifest that captures the full provenance of data, code, and configuration, often aligning with W3C PROV principles to enable rigorous traceability and reproducibility [250].

4.3.2 MLOps-aware lifecycles

The software lifecycle is increasingly extended to incorporate ML-specific processes, giving rise to a "continuous science" approach. Experiment tracking systems automatically record hyperparameters, data hashes, random seeds, metrics, and model artifacts, ensuring that every run is fully auditable and reproducible. Candidate models are promoted to registries with semantic versioning and are accompanied by Model Cards that document their intended use, performance characteristics, and ethical considerations [61]. Classical DevOps pipelines are similarly extended with continuous integration, delivery, and training (CI/CD/CT) capabilities. These pipelines incorporate continuous training triggers, model drift detectors, and shadow deployments, enabling the safe rollout of retrained models when data distributions evolve [251, 252]. Once deployed, operational monitoring expands beyond traditional measures of system uptime and latency to include data quality checks—such as monitoring for missingness or schema violations—real-time model performance on evaluation slices, and fairness or robustness indicators. In the event of detected regressions, incident response playbooks specify immediate rollback to previous model versions, minimizing the risk of propagating errors in production [218].

While the discussion above draws heavily from mature industrial MLOps stacks, most academic scientific environments operate under fundamentally different constraints. Research laboratories typically have small development teams, reliance on heterogeneous and sometimes outdated HPC environments, and budgets tied to 2–3-year grant cycles that rarely support long-term DevOps engineering roles. Continuous deployment, automated retraining, feature stores, and 24/7 monitoring may therefore be aspirational rather than feasible [253, 254].

An emerging distinction is thus between industrial-grade MLOps, emphasizing reliability at scale, operational risk management, and ongoing maintenance, and academic-grade MLOps, which prioritizes recomputability, transparent provenance, low-cost automation, and knowledge transfer across student turnover. Academic-grade workflows tend to benefit most from lightweight components: version-controlled experiment tracking, containerized execution environments, and periodic—not continuous—retraining schedules [255, 256]. Cloud-based managed services such as GitHub Actions, GitLab CI, and low-cost experiment trackers (e.g., Weights & Biases academic tiers, MLflow on institutional clusters) help reduce infrastructure burden while preserving visibility into the full model lifecycle [257, 258]. In many cases, "minimum viable MLOps" aligned with FAIR principles—clear metadata, accessible artifacts, and documented pipelines—is sufficient to ensure a scientific software system can be understood, reused, and extended after the original project ends [254, 255]. Developing a shared vocabulary for academic-grade MLOps may help align software sustainability expectations between domain scientists, grant agencies, and research software engineers, enabling reproducibility without imposing industry-scale operational complexity.

4.4 Testing and verification for data and models

Testing AI-enabled scientific software must extend far beyond simple unit tests; it needs to cover data integrity, model behavior, reproducibility, and robustness under changing conditions. Without such a thorough testing and verification regime, scientific conclusions risk being fragile, untrustworthy, or non-reproducible.

A reimagined "testing pyramid" for AI systems begins with verification at the data and software component level and builds up to more global model and system tests. At the base are unit and property tests: these verify correctness of data transformations, feature-engineering routines, and numerical stability under edge cases (for example, ensuring scaling, normalization, or interpolation functions behave well when confronted with missing or extreme values). In parallel, schema contracts and data validation checks must be enforced at pipeline boundaries to ensure that inputs and outputs maintain expected formats, distributions, and statistical properties (such as ranges, missingness, and label balance).

At the model level, verification includes deterministic "smoke tests" in which seeded runs are used to confirm basic correctness and detect glaring issues (such as exploding gradients or failures to converge). Further up, metamorphic testing strategies are essential where exact oracles are unavailable: if certain input transformations should lead to predictable output relationships (such as invariance, monotonicity, or symmetry), metamorphic relations can be used to detect faults. An example is the recent study by Reichert et al. [259], which shows how metamorphic testing can uncover differences in model behavior even when traditional calibration/validation passed. Regression tests using fixed evaluation sets also guard against unintended degradations of performance after refactoring or retraining.

Robustness and distribution-shift testing are also indispensable. Scientific models frequently face scenarios outside their original training regime: shifts in input distributions, unseen subpopulations, or perturbed environmental conditions. Evaluations should therefore report not just aggregate metrics, but also performance broken down by relevant subgroups and under explicit shifts. Documentation frameworks such as Model Cards recommend that model developers list limitations, intended use cases, and failure modes to help users understand where a model may or may not generalize. The work by Mitchell et al. [61] is a proposing approach for such structured documentation.

Finally, reproducibility checks should be built into both the development pipeline and the peer review process. Authors should publish code or scripts that regenerate figures, archive intermediate artifacts (datasets, feature transformations, model checkpoints), and specify seed settings and hardware details. The peer-review community is increasingly endorsing "artifact evaluation" tracks or badges to incentivize reproducible submissions. In related work, Liang et al. [260] examin how well model documentation practices currently fulfill transparency and reproducibility goals. Also, Amith et al. [261] propose ways to ensure model documentation is computable, structured, and machine-interpretable.

5 A pragmatic checklist (adaptable to most labs)

To promote reproducibility and transparency, research workflows should be designed with recomputation in mind - see Figure 6. This requires publishing detailed environment specifications, such as Conda configurations paired with container images, and supplementing them with tools like ReproZip or OCI images to safeguard against long-tail dependencies. Beyond environment management, computational pipelines should be made declarative. Frameworks such as Snakemake or Nextflow, or their equivalents, allow each node to explicitly define its environment and input–output schema. These directed acyclic graphs (DAGs) should be committed to version control to ensure consistency and traceability over time.

Equally critical is the systematic tracking of experiments and data. Datasets should be hashed, and experimental seeds and metrics rigorously logged to guarantee reproducibility across runs. Model releases should be promoted through a registry that includes standardized documentation, such as Model Cards, to capture intended use and performance characteristics. Once pipelines and data management are in place, testing must extend beyond simple validation. Property-based and metamorphic testing, robustness evaluations, and regression testing should be systematically employed. Continuous integration pipelines can be used to automatically execute notebooks and regenerate results, ensuring that updates do not compromise earlier findings.

Finally, responsible research demands proactive governance of risk. This entails maintaining an AI risk register and aligning controls with established frameworks such as the NIST AI Risk Management Framework. Where relevant, compliance with regulatory frameworks, such as the European Union's AI Act, should also be ensured. Mapping obligations to these categories not only mitigates risks but also fosters trustworthiness and accountability in research practices.

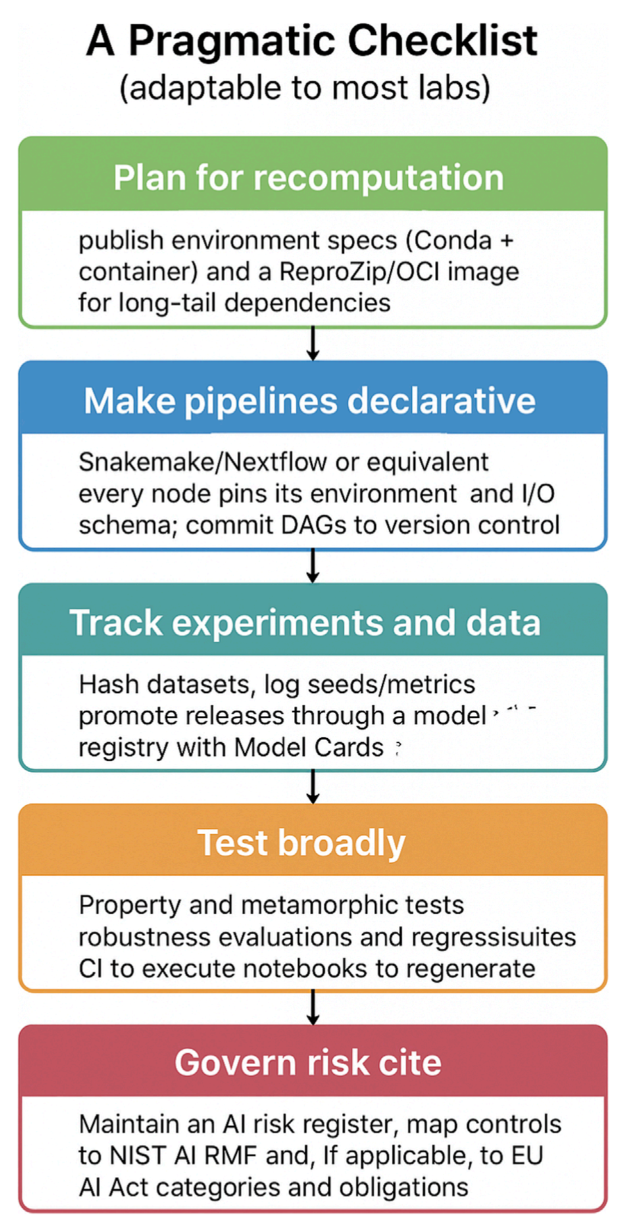
A Pragmatic Checklist
(adaptable to most labs)
Plan for recomputation
publish environment specs (Conda + container) and a ReproZip/OCI image for long-tail dependencies
Make pipelines declarative
Snakemake/Nextflow or equivalent every node pins its environment and I/O schema; commit DAGs to version control
Track experiments and data
Hash datasets, log seeds/metrics promote releases through a model registry with Model Cards
Test broadly
Property and metamorphic tests robustness evaluations and regressisuites CI to execute notebooks to regenerate
Govern risk cite
Maintain an AI risk register, map controls to NIST AI RMF and, If applicable, to EU AI Act categories and obligations

FIGURE 6
A pragmatic checklist for reproducible and responsible research. The checklist outlines five key practices adaptable to most laboratory settings: planning for recomputation through published environment specifications; making computational pipelines declarative and version-controlled; systematically tracking experiments and datasets; conducting broad testing, including robustness and regression evaluations; and governing risk through structured registers and compliance with established frameworks.

5.1 Practical blueprint: building an AI-era soft-matter stack

The development of an AI-era soft-matter simulation stack involves a structured, multi-stage workflow that integrates computational modeling with machine learning. Figure 7 illustrates a structured workflow for constructing an AI-driven soft-matter modeling stack, integrating multiscale simulations, machine learning, and uncertainty quantification. The process begins with problem framing, where the appropriate resolution—ranging from atomistic to coarse-grained to field-level representations—is selected according to the target observables. This step ensures that subsequent modeling efforts are aligned with the scientific questions of interest. Following this, a data strategy is established, which involves curating initial seed datasets, defining active-learning

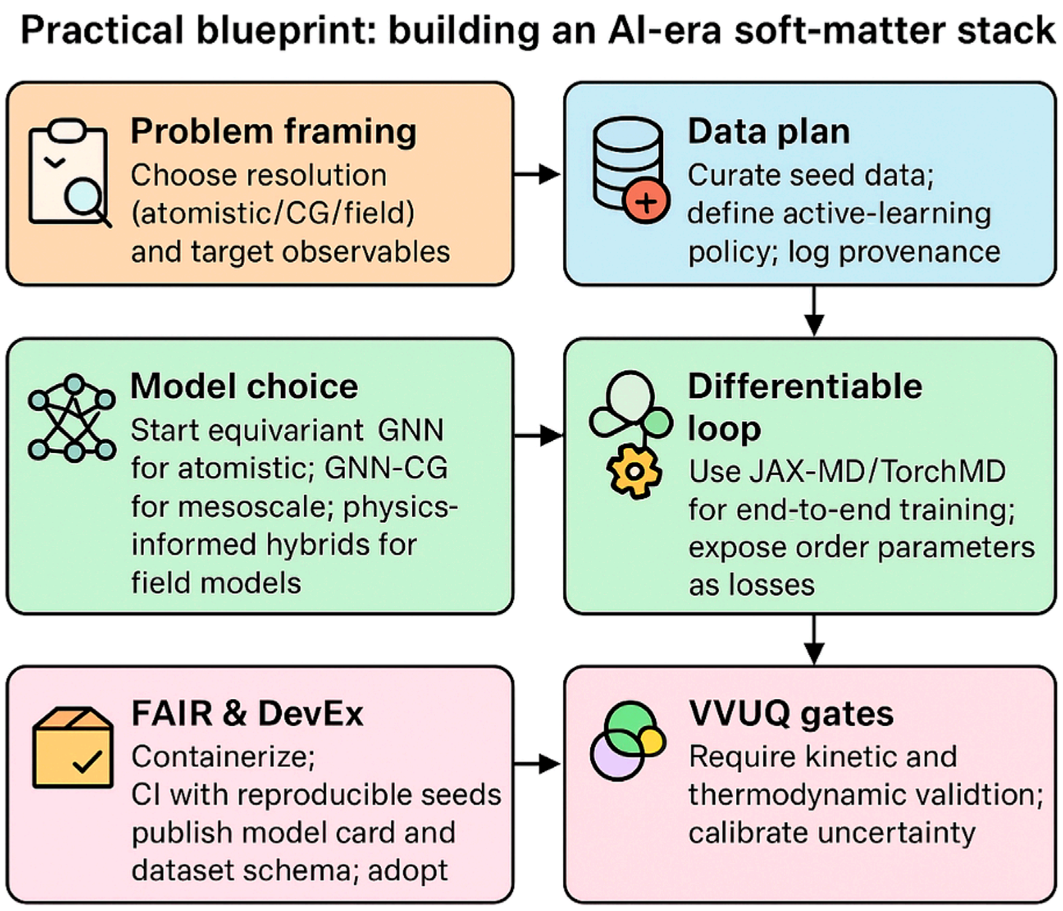
Practical blueprint: building an AI-era soft-matter stack
Problem framing
Choose resolution (atomistic/CG/field) and target observables
Data plan
Curate seed data; define active-learning policy; log provenance
Model choice
Start equivariant GNN for atomistic; GNN-CG for mesoscale; physics-informed hybrids for field models
Differentiable loop
Use JAX-MD/TorchMD for end-to-end training; expose order parameters as losses
FAIR & DevEx
Containerize; CI with reproducible seeds publish model card and dataset schema; adopt
VVUQ gates
Require kinetic and thermodynamic validtion; calibrate uncertainty

FIGURE 7
Practical blueprint for constructing an AI-era soft-matter simulation stack. The workflow progresses through six key stages: (1) Problem framing—selecting resolution and target observables; (2) Data plan—curating seed data, defining active-learning policies, and logging provenance; (3) Model choice—deploying equivariant GNNs for atomistic scales, GNN-CG for mesoscale, and physics-informed hybrids for field models; (4) Differentiable loop—enabling end-to-end training with JAX-MD/TorchMD and exposing order parameters as losses; (5) VVUQ gates—validating kinetics and thermodynamics while calibrating uncertainties; and (6) FAIR and DevEx—ensuring reproducibility, containerization, and compliance with FAIR4RS guidelines.

policies to iteratively improve model performance, and maintaining detailed provenance records to guarantee reproducibility.

The modeling stage leverages modern machine learning architectures tailored to the system scale: equivariant graph neural networks are employed for atomistic systems, coarse-grained GNNs for mesoscale representations, and hybrid physics-informed networks for continuum or field-level models. These models are integrated within a differentiable training loop, often implemented using frameworks such as JAX-MD or TorchMD, which allows order parameters and physical observables to be directly exposed as loss functions, enabling end-to-end optimization of both predictive accuracy and physical consistency.

To ensure reliability, the workflow incorporates verification, validation, and uncertainty quantification (VVUQ) gates. These steps evaluate both kinetic and thermodynamic predictions, calibrate uncertainties, and provide confidence in the model's predictive capabilities. Finally, the stack is designed with FAIR principles and developer experience in mind, including containerization for portability, continuous integration with reproducible seeds, comprehensive documentation via model cards and dataset schemas, and adherence to FAIR4RS guidelines to promote transparency and reusability within the scientific community. Collectively, this structured approach provides a practical blueprint for the systematic design, training, and deployment of AI-driven soft-matter simulations.

6 Conclusion

Scientific software is undergoing a profound transformation in the era of AI. The traditional priorities of correctness, performance, and reproducibility must now coexist with the demands of data-driven modeling, continuous retraining, and governance under evolving regulatory frameworks. This work has argued that meeting these demands requires a lifecycle-oriented approach in which reproducibility, provenance, and risk management are designed into the software from the outset rather than added as afterthoughts. Soft matter physics, with its inherently multiscale nature, rugged free-energy landscapes, and rich experimental interfaces, provides an ideal proving ground for these ideas. In this context, machine-learned interatomic and coarse-grained potentials, differentiable simulation frameworks, and closed-loop inverse design workflows offer not just improved accuracy but a fundamentally new mode of scientific practice—one in which modeling, data generation, and design are integrated into adaptive, end-to-end pipelines.

A central idea is that reproducibility must be treated as an architectural property of scientific software. Dataset versioning, content-addressable storage, and environment capture through containers or computational capsules ensure that results remain stable and recomputable even as dependencies evolve. Declarative workflows enable simulations, training steps, and evaluation routines to be specified in a portable, inspectable form, while experiment tracking and model registries preserve provenance across model updates and retraining cycles. When these practices are adopted systematically, they reduce technical debt, minimize the risk of silent errors, and make research outputs durable and trustworthy over time.

Equally important is the integration of governance into the scientific software lifecycle. As machine learning models increasingly mediate scientific inference, they also introduce new modes of failure and new obligations for transparency. Governance frameworks such as the NIST AI Risk Management Framework and the EU AI Act provide a foundation for embedding accountability, monitoring, and human oversight directly into modeling pipelines. Scientific teams must embrace documentation that is both human-readable and machine-actionable, capturing intended use, performance characteristics, limitations, and known risks in model cards and dataset datasheets. Continuous monitoring for model drift, robust evaluation under distributional shifts, and well-defined rollback procedures are all part of a responsible approach to AI-enabled science.

The future agenda for AI-augmented scientific software must therefore focus on building infrastructure that is as auditable as it is performant. Living benchmarks and shared testbeds will be critical for enabling fair comparison and cumulative progress across the community. Executable publications and artifact evaluation processes should become routine, ensuring that results can be regenerated by reviewers and future researchers alike. Machine learning pipelines should become debt-aware by default, incorporating tools that detect stale dependencies, configuration drift, and potential data leakage before they compromise results. Finally, as large language models and other generative systems increasingly assist in code generation, data curation, and analysis, there is a need to establish norms for attribution, verification, and human oversight so that scientific accountability is preserved.

If these principles are embraced, the scientific community can turn the current fragility of AI-enabled research into a foundation for robust, cumulative knowledge generation. Rather than being a source of reproducibility crises, machine learning can become a catalyst for more transparent, reproducible, and adaptive science. Soft matter physics, by virtue of its multiscale complexity and its appetite for high-fidelity yet computationally efficient modeling, stands at the forefront of this transformation. In positioning itself not just as a beneficiary but as a testbed for AI-era software practices, the field can help define a new standard for scientific software: one that is not only correct and performant but also recomputable, auditable, and aligned with the growing societal expectations for trustworthy and accountable AI.

Author contributions

NC: Conceptualization, Data curation, Formal Analysis, Funding acquisition, Investigation, Methodology, Project administration, Resources, Software, Supervision, Validation, Visualization, Writing – original draft, Writing – review and editing.

Funding

The authors declare that no financial support was received for the research and/or publication of this article.

Conflict of interest

The author declares that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Generative AI statement

The authors declare that no Generative AI was used in the creation of this manuscript.

Any alternative text (alt text) provided alongside figures in this article has been generated by Frontiers with the support of artificial intelligence and reasonable efforts have been made to ensure accuracy, including review by the authors wherever possible. If you identify any issues, please contact us.

Publisher's note

References

1. Helfrich W. Elastic properties of lipid bilayers: theory and possible experiments. *Z Naturforsch C* (1973) 28:693–703. doi:10.1515/znc-1973-11-1209

2. Buff FP, Lovett RA, Stillinger FH. Interfacial density profile for fluids in the critical region. *Phys Rev Lett* (1965) 15:621–3. doi:10.1103/PhysRevLett.15.621

3. Aarts D, Schmidt M, Lekkerkerker HNW. Direct visual observation of thermal capillary waves. *Science* (2004) 304:847–50. doi:10.1126/science.1097116

4. Pusey PN, van Megen W. Observation of a glass transition in suspensions of spherical colloidal particles. *Phys Rev Lett* (1987) 59:2083–6. doi:10.1103/PhysRevLett.59.2083

5. Lu PJ, Zaccarelli E, Ciulla F, Schofield AB, Sciortino F, Weitz DA. Gelation of particles with short-range attraction. *Nature* (2008) 453:499–503. doi:10.1038/nature06931

6. Zaccarelli E. Colloidal gels: equilibrium and non-equilibrium routes. *J Phys Condensed Matter* (2007) 19:323101. doi:10.1088/0953-8984/19/32/323101

7. Español P, Warren P. Statistical mechanics of dissipative particle dynamics. *Europhysics Lett* (1995) 30:191–6. doi:10.1209/0295-5075/30/4/001

8. Likhtman AE, McLeish TCB. Quantitative theory for linear dynamics of linear entangled polymers. *Macromolecules* (2002) 35:6332–43. doi:10.1021/ma0200219

9. Dünweg B, Ladd AJC. Lattice boltzmann simulations of soft matter systems. In: C Holm, K Kremer, editors. *Advanced computer simulation approaches for soft matter sciences III*. Berlin, Heidelberg: Springer Berlin Heidelberg. 89–166. doi:10.1007/978-3-540-87706-6_2

10. Izvekov S, Voth GA. A multiscale coarse-graining method for biomolecular systems. *J Phys Chem B* (2005) 109:2469–73. doi:10.1021/jp044629q

11. Murtola T, Bunker A, Vattulainen I, Deserno M, Karttunen M. Multiscale modeling of emergent materials: biological and soft matter. *Phys Chem Chem Phys* (2009) 11:1869–92. doi:10.1039/B818051B

12. Noid WG. Perspective: coarse-Grained models for biomolecular systems. *The J Chem Phys* (2013) 139:090901. doi:10.1063/1.4818908

13. Brini E, Algaer EA, Ganguly P, Li C, Rodríguez-Ropero F, van der Vegt NFA. Systematic coarse-graining methods for soft matter simulations – a review. *Soft Matter* (2013) 9:2108–19. doi:10.1039/C2SM27201F

14. Guenza MG, Dinpajooh M, McCarty J, Lyubimov IY. Accuracy, transferability, and efficiency of coarse-grained models of molecular liquids. *J Phys Chem B* (2018) 122:10257–78. doi:10.1021/acs.jpcb.8b06687

15. Kirkpatrick S, Gelatt CD, Vecchi MP. Optimization by simulated annealing. *Science* (1983) 220:671–80. doi:10.1126/science.220.4598.671

16. Frenkel D, Smit B, Ratner MA. Understanding molecular simulation: from algorithms to applications. *Phys Today* (1997) 50:66. doi:10.1063/1.881812

17. Wales DJ. Energy landscapes: calculating pathways and rates. *Int Rev Phys Chem* (2006) 25:237–82. doi:10.1080/01442350600676921

18. Miskin MZ, Jaeger HM. Adapting granular materials through artificial evolution. *Nat Mater* (2013) 12:326–31. doi:10.1038/nmat3543

19. Gómez-Bombarelli R, Wei JN, Duvenaud D, Hernández-Lobato JM, Sánchez-Lengeling B, Sheberla D, et al. Automatic chemical design using a data-driven continuous representation of molecules. *ACS Cent Sci* (2018) 4:268–76. doi:10.1021/acscentsci.7b00572

20. Behler J, Parrinello M. Generalized neural-network representation of high-dimensional potential-energy surfaces. *Phys Rev Lett* (2007) 98:146401. doi:10.1103/PhysRevLett.98.146401

21. Bartók AP, Payne MC, Kondor R, Csányi G. Gaussian approximation potentials: the accuracy of quantum Mechanics, without the electrons. *Phys Rev Lett* (2010) 104:136403. doi:10.1103/PhysRevLett.104.136403

22. Unke OT, Meuwly M. PhysNet: a neural network for predicting energies, forces, dipole moments, and partial charges. *J Chem Theor Comput* (2019) 15:3678–93. doi:10.1021/acs.jctc.9b00181

23. Schütt KT, Arbabzadah F, Chmiela S, Müller KR, Tkatchenko A. Quantum-chemical insights from deep tensor neural networks. *Nat Commun* (2017) 8:13890. doi:10.1038/ncomms13890

24. Zhang L, Han J, Wang H, Car R. Deep potential molecular dynamics: a scalable model with the accuracy of quantum mechanics. *Phys Rev Lett* (2018) 120:143001. doi:10.1103/PhysRevLett.120.143001

25. Chmiela S, Tkatchenko A, Sauceda HE, Poltavsky I, Schütt KT, Müller K-R. Machine learning of accurate energy-conserving molecular force fields. *Sci Adv* (2017) 3:e1603015. doi:10.1126/sciadv.1603015

26. Smith JS, Isayev O, Roitberg AE. ANI-1: an extensible neural network potential with DFT accuracy at force field computational cost. *Chem Sci* (2017) 8:3192–203. doi:10.1039/C6SC05720A

27. Thomas N, Smidt T, Kearnes S, Yang L, Li L, Kohlhoff K, et al. Tensor field networks: rotation- and translation-equivariant neural networks for 3D point clouds. (2018). doi:10.48550/arXiv.1802.08219

28. Schütt KT, Kindermans P-J, Sauceda HE, Chmiela S, Tkatchenko A, Müller K-R. SchNet: a continuous-filter convolutional neural network for modeling quantum interactions. In: *Proceedings of the 31st international conference on neural information processing systems NIPS'17*. Red Hook, NY, USA: Curran Associates Inc. 992–1002.

29. Anderson B, Hy T-S, Kondor R. Cormorant: covariant molecular neural networks. In: *Proceedings of the 33rd international conference on neural information processing systems*. Red Hook, NY, USA: Curran Associates Inc.

30. Thölke P, De Fabritiis G. TorchMD-NET: Equivariant transformers for neural network based molecular potentials. arXiv e-prints (2022). doi:10.48550/arXiv.2202.02541

31. Batzner S, Musaelian A, Sun L, Geiger M, Mailoa JP, Kornbluth M, et al. E(3)-equivariant graph neural networks for data-efficient and accurate interatomic potentials. *Nat Commun* (2022) 13:2453. doi:10.1038/s41467-022-29939-5

32. Batatia I, Péter Kovács D, Simm GNC, Ortner C, Csányi G. MACE: higher order equivariant message passing neural networks for fast and accurate force fields. arXiv:2206 (2022). doi:10.48550/arXiv.2206.07697

33. Musaelian A, Batzner S, Johansson A, Sun L, Owen CJ, Kornbluth M, et al. Learning local equivariant representations for large-scale atomistic dynamics. *Nat Commun* (2023) 14:579. doi:10.1038/s41467-023-36329-y

34. Husic BE, Charron NE, Lemm D, Wang J, Pérez A, Majewski M, et al. Coarse graining molecular dynamics with graph neural networks. *The J Chem Phys* (2020) 153:194101. doi:10.1063/5.0026133

35. Noé F, Tkatchenko A, Müller K-R, Clementi C. Machine learning for molecular simulation. *Annu Rev Phys Chem* (2020) 71:361–90. doi:10.1146/annurev-physchem-042018-052331

36. Schoenholz S, Cubuk ED, Jax MD. A framework for differentiable physics. In: H Larochelle, M Ranzato, R Hadsell, MF Balcan, H Lin, editors. *Advances in neural information processing systems*. Curran Associates, Inc. 11428–41. Available online at: https://proceedings.neurips.cc/paper_files/paper/2020/file/83d3d4b6c9579515e1679aca8cbc8033-Paper.pdf (Accessed November 18, 2025).

37. Doerr S, Majewski M, Pérez A, Krämer A, Clementi C, Noe F, et al. TorchMD: a deep learning framework for molecular simulations. *J Chem Theor Comput* (2021) 17:2355–63. doi:10.1021/acs.jctc.0c01343

38. Greener JG. Differentiable simulation to develop molecular dynamics force fields for disordered proteins. *Chem Sci* (2024) 15:4897–909. doi:10.1039/D3SC05230C

39. Innes M, Edelman A, Fischer K, Rackauckas C, Saba E, Shah VB, et al. A differentiable programming system to bridge machine learning and scientific computing. (2019). doi:10.48550/arXiv.1907.07587

40. Vargas-Hernández RA, Jorner K, Pollice R, Aspuru-Guzik A. Inverse molecular design and parameter optimization with hückel theory using automatic differentiation. *J Chem Phys* (2023) 158:104801. doi:10.1063/5.0137103

41. Lamprecht A-L, Garcia L, Kuzak M, Martinez C, Arcila R, Martin Del Pico E, et al. Towards FAIR principles for research software. *Data Sci* (2020) 3:37–59. doi:10.3233/DS-190026

42. Jiménez RC, Kuzak M, Alhamdoosh M, Barker M, Batut B, Borg M, et al. Four simple recommendations to encourage best practices in research software. *F1000Res* (2017) 6:ELIXIR-876–. doi:10.12688/f1000research.11407.1

43. Barker M, Chue Hong NP, Katz DS, Lamprecht A-L, Martinez-Ortiz C, Psomopoulos F, et al. Introducing the FAIR principles for research software. *Scientific Data* (2022) 9:622. doi:10.1038/s41597-022-01710-x

44. Kreuzberger D, Kühl N, Hirschl S. Machine learning operations (MLOps): overview, definition, and architecture. (2022). doi:10.48550/arXiv.2205.02302

45. Testi M, Ballabio M, Frontoni E, Iannello G, Moccia S, Soda P, et al. MLOps: a taxonomy and a methodology. *IEEE Access* (2022). 10:63606–18. doi:10.1109/ACCESS.2022.3181730

46. Symeonidis G, Nerantzis E, Kazakis A, Papakostas GA. MLOps – definitions, tools and challenges (2022). 60. doi:10.1109/ccwc54503.2022.9720902

47. Baker M. 1,500 scientists lift the lid on reproducibility. *Nature* (2016) 533:452–4. doi:10.1038/533452a

48. Wilson G, Bryan J, Cranston K, Kitzes J, Nederbragt L, Teal TK. Good enough practices in scientific computing. *PLOS Comput Biol* (2017) 13:e1005510. doi:10.1371/journal.pcbi.1005510

49. Sandve GK, Nekrutenko A, Taylor J, Hovig E. Ten simple rules for reproducible computational research. *PLOS Comput Biol* (2013) 9:e1003285. doi:10.1371/journal.pcbi.1003285

50. Frieder S, Pinchetti L, Chevalier A, Griffiths R-R, Salvatori T, Lukasiewicz T, et al. Mathematical capabilities of ChatGPT. In: *Proceedings of the 37th international conference on neural information processing systems*. New Orleans, LA, USA: Curran Associates Inc.

51. OpenAI AJ, Adler S, Agarwal S, Ahmad L, Akkaya I, Leoni Aleman F, et al. GPT-4 technical report. (2023). doi:10.48550/arXiv.2303.08774

52. Roychoudhury A, Pasareanu C, Pradel M, Ray B. Agentic AI software engineers: programming with trust. (2025). doi:10.48550/arXiv.2502.13767

53. Souza R, Gueroudji A, DeWitt S, Rosendo D, Ghosal T, Ross R, et al. PROV-AGENT: unified provenance for tracking AI agent interactions in agentic workflows. (2025) 467, 73. doi:10.1109/escience65000.2025.00093

54. Souza R, Azevedo L, Lourenço V, Soares E, Thiago R, Brandão R, et al. Provenance data in the machine learning lifecycle in computational science and engineering. In *2019 IEEE/ACM workflows in support of large-scale science (WORKS)*. 1–10. doi:10.1109/WORKS49585.2019.00006

55. Bose R, Frew J. Lineage retrieval for scientific data processing: a survey. *ACM Comput Surv* (2005) 37:1–28. doi:10.1145/1057977.1057978

56. Pearce H, Ahmad B, Tan B, Dolan-Gavitt B, Karri R. Asleep at the keyboard? Assessing the security of GitHub copilot's code contributions. (2021). doi:10.48550/arXiv.2108.09293

57. Atemkeng M, Hamlomo S, Welman B, Oyentunji N, Ataei P, Fendji J. Ethics of software programming with generative AI: is programming without generative AI always radical? (2024). doi:10.48550/arXiv.2408.10554

58. Voinea DV. Ai and copyright - who owns ai generated content? *Social Sci Education Res Rev* (2023). 10:262–7. doi:10.5281/zenodo.15252004

59. Bhatia A, Khomh F, Adams B, E Hassan A. An empirical study of self-admitted technical debt. *Machine Learn Softw* (2023). doi:10.48550/arXiv.2311.12019

60. Hassan AE, Li H, Lin D, Adams B, Chen T-H, Kashiwa Y, et al. Agentic software engineering: foundational pillars and a research roadmap (2025). doi:10.48550/arXiv.2509.06216

61. Mitchell M, Wu S, Zaldivar A, Barnes P, Vasserman L, Hutchinson B, et al. Model cards for model reporting. In: *Proceedings of the conference on fairness, accountability, and transparency*. p. 220–9. doi:10.1145/3287560.3287596

62. Bommasani R, Hudson DA, Adeli E, Altman R, Arora S, Arx S, et al. On the opportunities and risks of foundation models. (2022). doi:10.48550/arXiv.2108.07258

63. Peng RD. Reproducible research in computational science. *Science* (2011) 334:1226–7. doi:10.1126/science.1213847

64. Stodden V, Seiler J, Ma Z. An empirical analysis of journal policy effectiveness for computational reproducibility. *Proc Natl Acad Sci* (2018) 115:2584–9. doi:10.1073/pnas.1708290115

65. Smith JS, Zubatyuk R, Nebgen B, Lubbers N, Barros K, Roitberg AE, et al. The ANI-1ccx and ANI-1x data sets, coupled-cluster and density functional theory properties for molecules. *Scientific Data* (2020) 7:134. doi:10.1038/s41597-020-0473-z

66. Wilkinson MD, Dumontier M, Aalbersberg IJ, Appleton G, Axton M, Baak A, et al. The FAIR guiding principles for scientific data management and stewardship. *Scientific Data* (2016) 3:160018. doi:10.1038/sdata.2016.18

67. Pineau J, Vincent-Lamarre P, Sinha K, Larivière V, Beygelzimer A, d'Alché-Buc F, et al. Improving reproducibility in machine learning research (A report from the NeurIPS 2019 reproducibility program). arXiv e-prints (2020). doi:10.48550/arXiv.2003.12206

68. Sculley D, Holt G, Golovin D, Davydov E, Phillips T, Ebner D, et al. Hidden technical debt in machine learning systems. In: C Cortes, N Lawrence, D Lee, M Sugiyama, R Garnett, editors. *Advances in neural information processing systems*. New York, NY: Curran Associates, Inc. Available online at: https://proceedings.neurips.cc/paper_files/paper/2015/file/86df7dcfd896fcaf2674f757a2463eba-Paper.pdf (Accessed November 18, 2025).

69. Arpteg A, Brinne B, Crnkovic-Friis L, Bosch J. Software engineering challenges of deep learning. In: *2018 44th euromicro conference on software engineering and advanced applications (SEAA)*. 50–9. doi:10.1109/SEAA.2018.00018

70. Amershi S, Begel A, Bird C, DeLine R, Gall H, Kamar E, et al. Software engineering for machine learning: a case study. In: *2019 IEEE/ACM 41st international conference on software engineering: software engineering in practice (ICSE-SEIP)*. p. 291–300. doi:10.1109/ICSE-SEIP.2019.00042

71. Ensign D, Friedler SA, Neville S, Scheidegger C, Venkatasubramanian S. Runaway feedback loops in predictive policing. arXiv e-prints (2017). doi:10.48550/arXiv.1706.09847

72. Raji ID, Smart A, White RN, Mitchell M, Gebru T, Hutchinson B, et al. Closing the AI accountability gap: defining an end-to-end framework for internal algorithmic auditing. *arXiv E-Prints* (2020). doi:10.48550/arXiv.2001.00973

73. Barocas S, Hardt M, Narayanan A. *Fairness and machine learning: limitations and opportunities*. MIT Press (2023).

74. Barberis A, Aerts HJWL, Buffa FM. Robustness and reproducibility for AI learning in biomedical sciences: RENOIR. *Scientific Rep* (2024) 14:1933. doi:10.1038/s41598-024-51381-4

75. Paleyes A, Urma R-G, Lawrence ND. Challenges in deploying machine learning: a survey of case studies. arXiv e-prints (2020). doi:10.48550/arXiv.2011.09926

76. Wilson G, Aruliah DA, Brown CT, Chue Hong NP, Davis M, Guy RT, et al. Best practices for scientific computing. *PLOS Biol* (2014) 12. doi:10.1371/journal.pbio.1001745

77. Wilson G. Software carpentry: lessons learned. arXiv e-prints (2013). doi:10.48550/arXiv.1307.5448

78. Stodden V, Leisch F, Peng RD. *Implementing reproducible research*. Boca Raton, FL: CRC Press (2014).

79. Hutson M. Artificial intelligence faces reproducibility crisis. *Science* (2018) 359:725–6. doi:10.1126/science.359.6377.725

80. Cheimarios N, Kokkoris G, Boudouvis AG. Multiscale modeling in chemical vapor deposition processes: models and methodologies. *Arch Comput Methods Eng* (2021) 28:637–72. doi:10.1007/s11831-019-09398-w

81. Wang J, Olsson S, Wehmeyer C, Pérez A, Charron NE, de Fabritiis G, et al. Machine learning of coarse-grained molecular dynamics force fields. *ACS Cent Sci* (2019) 5:755–67. doi:10.1021/acscentsci.8b00913

82. Batatia I, Batzner S, Kovács DP, Musaelian A, Simm GNC, Drautz R, et al. The design space of E(3)-equivariant atom-centred interatomic potentials. *Nat Machine Intelligence* (2025) 7:56–67. doi:10.1038/s42256-024-00956-x

83. Bolhuis PG, Dellago C. Trajectory-based rare event simulations. In: *Reviews in computational chemistry reviews in computational chemistry*. 111–210. doi:10.1002/9780470890905.ch3

84. Tiwary P, Parrinello M. From metadynamics to dynamics. *Phys Rev Lett* (2013) 111:230602. doi:10.1103/PhysRevLett.111.230602

85. Sidky H, Chen W, Ferguson AL. Molecular latent space simulators. *Chem Sci* (2020) 11:9459–67. doi:10.1039/D0SC03635H

86. Cheimarios N, To D, Kokkoris G, Memos G, Boudouvis AG. Monte carlo and kinetic monte carlo models for deposition processes: a review of recent works. *Front Phys* (2021) 9:631918. doi:10.3389/fphy.2021.631918

87. Röding M, Tomaszewski P, Yu S, Borg M, Rönnols J. Machine learning-accelerated small-angle X-ray scattering analysis of disordered two- and three-phase materials. *Front Mater* (2022) 9:956839. doi:10.3389/fmats.2022.956839

88. Horwath JP, Lin X-M, He H, Zhang Q, Dufresne EM, Chu M, et al. AI-NERD: elucidation of relaxation dynamics beyond equilibrium through AI-informed X-ray photon correlation spectroscopy. *Nat Commun* (2024) 15:5945. doi:10.1038/s41467-024-49381-z

89. Weigert M, Schmidt U, Boothe T, Müller A, Dibrov A, Jain A, et al. Content-aware image restoration: pushing the limits of fluorescence microscopy. *Nat Methods* (2018) 15:1090–7. doi:10.1038/s41592-018-0216-7

90. Ouyang W, Aristov A, Lelek M, Hao X, Zimmer C. Deep learning massively accelerates super-resolution localization microscopy. *Nat Biotechnol* (2018) 36:460–8. doi:10.1038/nbt.4106

91. Speiser A, Müller L-R, Hoess P, Matti U, Obara CJ, Legant WR, et al. Deep learning enables fast and dense single-molecule localization with high accuracy. *Nat Methods* (2021) 18:1082–90. doi:10.1038/s41592-021-01236-x

92. Midtvedt B, Helgadottir S, Argun A, Pineda J, Midtvedt D, Volpe G. Quantitative digital microscopy with deep learning. *Appl Phys Rev* (2021) 8:011310. doi:10.1063/5.0034891

93. Bepler T, Morin A, Rapp M, Brasch J, Shapiro L, Noble AJ, et al. Positive-unlabeled convolutional neural networks for particle picking in cryo-electron micrographs. *Nat Methods* (2019) 16:1153–60. doi:10.1038/s41592-019-0575-8

94. Wagner T, Merino F, Stabrin M, Moriya T, Antoni C, Apelbaum A, et al. SPHIRE-crYOLO is a fast and accurate fully automated particle picker for cryo-EM. *Commun Biol* (2019) 2:218. doi:10.1038/s42003-019-0437-z

95. MacLeod BP, Parlane FGL, Morrissey TD, Häse F, Roch LM, Dettelbach KE, et al. Self-driving laboratory for accelerated discovery of thin-film materials. *Sci Adv* (2020) 6:eaaz8867. doi:10.1126/sciadv.aaz8867

96. Yager KG, Majewski PW, Noack MM, Fukuto M. Autonomous x-ray scattering. *Nanotechnology* (2023) 34:322001. doi:10.1088/1361-6528/acd25a

97. Lee J, Shin S, Kim T, Park B, Choi H, Lee A, et al. Physics informed neural networks for fluid flow analysis with repetitive parameter initialization. *Sci Rep* (2025) 15:16740. doi:10.1038/s41598-025-99354-5

98. Karniadakis GE, Kevrekidis IG, Lu L, Perdikaris P, Wang S, Yang L. Physics-informed machine learning. *Nat Rev Phys* (2021) 3:422–40. doi:10.1038/s42254-021-00314-5

99. Kovachki N, Li Z, Liu B, Azizzadenesheli K, Bhattacharya K, Stuart A, et al. Neural operator: learning maps between function spaces. (2024). doi:10.5555/3648699.3648788

100. Hu S, Jin Q, Gao C, Zhang X, Lu M, He Y, et al. The new paradigm of computational fluid dynamics: empowering computational fluid dynamics with machine learning. *Phys Fluids* (2025) 37:081302. doi:10.1063/5.0280743

101. Jiang Z, Jiang J, Yao Q, Yang G. A neural network-based PDE solving algorithm with high precision. *Scientific Rep* (2023) 13:4479. doi:10.1038/s41598-023-31236-0

102. Um K, Brand R, Yun F, Holl P, Thuerey N. Solver-in-the-Loop: learning from differentiable physics to interact with iterative PDE-solvers (2021). doi:10.48550/arXiv.2007.00016

103. Di Fiore F, Nardelli M, Mainini L. Active learning and bayesian optimization: a unified perspective to learn with a goal. *Arch Computat Methods Eng* (2024) 31:2985–3013. doi:10.1007/s11831-024-10064-z

104. Gundersen OE, Coakley K, Kirkpatrick C, Gil Y. Sources of irreproducibility in machine learning: a review (2023). doi:10.48550/arXiv.2204.07610

105. Semmelrock H, Ross-Hellauer T, Kopeinik S, Theiler D, Haberl A, Thalmann S, et al. Reproducibility in machine learning-based research: overview. *Barriers and Drivers* (2024). doi:10.48550/arXiv.2406.14325

106. Heil BJ, Hoffman MM, Markowetz F, Lee S-I, Greene CS, Hicks SC. Reproducibility standards for machine learning in the life sciences. *Nat Methods* (2021) 18:1132–5. doi:10.1038/s41592-021-01256-7

107. U.S. Department of Commerce, National Institute of Standards and Technology *Artificial intelligence risk management framework (AI RMF 1.0)*, 1270. Washington, DC: NIST Special Publication (2023).

108. European Commission. Proposal for a regulation laying Down harmonised rules on artificial intelligence (artificial intelligence act). *COM/2021/206 Final* (2021). Available online at: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:52021PC0206 (Accessed November 18, 2025).

109. Boettiger C. An introduction to docker for reproducible research, with examples from the R environment. arXiv E-Prints (2014). doi:10.48550/arXiv.1410.0846

110. Ayton GS, Noid WG, Voth GA. Multiscale modeling of biomolecular systems: in serial and in parallel. *Curr Opin Struct Biol* (2007) 17:192–8. doi:10.1016/j.sbi.2007.03.004

111. Peter C, Kremer K. Multiscale simulation of soft matter systems – from the atomistic to the coarse-grained level and back. *Soft Matter* (2009) 5:4357–66. doi:10.1039/B912027K

112. Harmandaris VA, Kremer K. Predicting polymer dynamics at multiple length and time scales. *Soft Matter* (2009) 5:3920–6. doi:10.1039/B905361A

113. Chandler D. *Introduction to modern statistical mechanics*. Oxford University Press (1987).

114. Dellago C, Bolhuis PG, Geissler PL. Transition path sampling methods. In: M Ferrario, G Ciccotti, K Binder, editors. *Computer simulations in condensed matter systems: from materials to chemical biology volume 1*. Berlin, Heidelberg: Springer Berlin Heidelberg. 349–91. doi:10.1007/3-540-35273-2_10

115. Ferguson AL. Machine learning and data science in soft materials engineering. *J Phys Condensed Matter* (2017) 30:043002. doi:10.1088/1361-648X/aa98bd

116. Ricci E, Vergadou N. Integrating machine learning in the coarse-grained molecular simulation of polymers. *J Phys Chem B* (2023) 127:2302–22. doi:10.1021/acs.jpcb.2c06354

117. Kulichenko M, Barros K, Lubbers N, Li YW, Messerly R, Tretiak S, et al. Uncertainty-driven dynamics for active learning of interatomic potentials. *Nat Comput Sci* (2023) 3:230–9. doi:10.1038/s43588-023-00406-5

118. Zaverkin V, Holzmüller D, Christiansen H, Errica F, Alesiani F, Takamoto M, et al. Uncertainty-biased molecular dynamics for learning uniformly accurate interatomic potentials. *Npj Comput Mater* (2024) 10:83. doi:10.1038/s41524-024-01254-1

119. Jung GS, Choi JY, Lee SM. Active learning of neural network potentials for rare events. *Digital Discov* (2024) 3(3):514–27. doi:10.1039/D3DD00216K

120. Kang K, Purcell TAR, Carbogno C, Scheffler M. Accelerating the training and improving the reliability of machine-learned interatomic potentials for strongly anharmonic materials through active learning. *Phys Rev Mater* (2025) 9:063801. doi:10.1103/PhysRevMaterials.9.063801

121. Coli GM, Boattini E, Filion L, Dijkstra M. Inverse design of soft materials *via* a deep learning–based evolutionary strategy. *Sci Adv* (2022) 8:eabj6731. doi:10.1126/sciadv.abj6731

122. Loeffler HH, Wan S, Klähn M, Bhati AP, Coveney PV. Optimal molecular design: generative active learning combining REINVENT with precise binding free energy ranking simulations. *J Chem Theor Comput* (2024) 20:8308–28. doi:10.1021/acs.jctc.4c00576

123. Liao V, Jayaraman A. Inverse design of block polymer materials with desired nanoscale structure and macroscale properties. *JACS Au* (2025) 5:2810–24. doi:10.1021/jacsau.5c00377

124. Azizzadenesheli K, Kovachki N, Li Z, Liu-Schiaffini M, Kossaifi J, Anandkumar A. Neural operators for accelerating scientific simulations and design. *Nat Rev Phys* (2024) 6:320–8. doi:10.1038/s42254-024-00712-5

125. Jha PK. From theory to application: a practical introduction to neural operators in scientific computing. arXiv e-prints (2025). doi:10.48550/arXiv.2503.05598

126. Bezgin DA, Buhendwa AB, Adams NA. JAX-Fluids: a fully-differentiable high-order computational fluid dynamics solver for compressible two-phase flows. *Computer Phys Commun* (2023) 282:108527. doi:10.1016/j.cpc.2022.108527

127. Shankar V, Chakraborty D, Viswanathan V, Maulik R. Differentiable turbulence: closure as a partial differential equation constrained optimization. *Phys Rev Fluids* (2025) 10:024605. doi:10.1103/PhysRevFluids.10.024605

128. Fan X, Liu X, Wang M, Wang J-X. Diff-FlowFSI: a GPU-optimized differentiable CFD platform for high-fidelity turbulence and FSI simulations. arXiv e-prints (2025). doi:10.48550/arXiv.2505.23940

129. Shukla K, Oommen V, Peyvan A, Penwarden M, Plewacki N, Bravo L, et al. Deep neural operators as accurate surrogates for shape optimization. *Eng Appl Artif Intelligence* (2024) 129:107615. doi:10.1016/j.engappai.2023.107615

130. Mader CA, Martins JRRA. Derivatives for time-spectral computational fluid dynamics using an automatic differentiation adjoint. *AIAA J* (2012) 50:2809–19. doi:10.2514/1.J051658

131. Gao Y, Wu Y, Xia J. Automatic differentiation based discrete adjoint method for aerodynamic design optimization on unstructured meshes. *Chin J Aeronautics* (2017) 30:611–27. doi:10.1016/j.cja.2017.01.009

132. Lye KO, Mishra S, Ray D, Chandrashekar P. Iterative surrogate model optimization (ISMO): an active learning algorithm for PDE constrained optimization with deep neural networks. *Computer Methods Appl Mech Eng* (2021) 374:113575. doi:10.1016/j.cma.2020.113575

133. Tripathi M, Kumar S, Desale YB, Pant RS. Active learning - CFD integrated surrogate-based framework for shape optimization of LTA systems. In: *AIAA AVIATION FORUM AND ASCEND 2024 AIAA aviation forum and ASCEND co-located conference proceedings*. American Institute of Aeronautics and Astronautics. doi:10.2514/6.2024-4120

134. Wu D, Niu R, Chinazzi M, Vespignani A, Ma Y-A, Yu R. Deep bayesian active learning for accelerating stochastic simulation. (2021). doi:10.48550/arXiv.2106.02770

135. Lee T, Kim J, Lee C. Turbulence control for drag reduction through deep reinforcement learning. *Phys Rev Fluids* (2023) 8:024604. doi:10.1103/PhysRevFluids.8.024604

136. Suárez P, Alcántara-Ávila F, Miró A, Rabault J, Font B, Lehmkuhl O, et al. Active flow control for drag reduction through multi-agent reinforcement learning on a turbulent cylinder at ReD = 3900. *Flow Turbul Combust* (2025) 115:3–27. doi:10.1007/s10494-025-00642-x

137. Wang W, Chu X. Optimised flow control based on automatic differentiation in compressible turbulent channel flows. *J Fluid Mech* (2025) 1011:A1. doi:10.1017/jfm.2025.304

138. Maruf MU, Kim S, Ahmad Z. Equivariant machine learning interatomic potentials with global charge redistribution. arXiv E-Prints (2025). doi:10.48550/arXiv.2503.17949

139. Ji Y, Liang J, Xu Z. Machine-learning interatomic potentials for long-range systems. arXiv E-Prints (2025). doi:10.48550/arXiv.2502.04668

140. Anstine DM, Isayev O. Machine learning interatomic potentials and long-range physics. *J Phys Chem A* (2023) 127:2417–31. doi:10.1021/acs.jpca.2c06778

141. Shen Z, Sosa RI, Lengiewicz J, Tkatchenko A, Bordas SPA. Machine learning surrogate models of many-body dispersion interactions in polymer melts. arXiv e-prints (2025). doi:10.48550/arXiv.2503.15149

142. Mishin Y. Machine-learning interatomic potentials for materials science. *Acta Materialia* (2021) 214. doi:10.1016/j.actamat.2021.116980

143. Yan K, Bohde M, Kryvenko A, Xiang Z, Zhao K, Zhu S, et al. A materials foundation model *via* hybrid invariant-equivariant architectures. arXiv e-prints (2025). doi:10.48550/arXiv.2503.05771

144. van der Oord C, Sachs M, Kovács DP, Ortner C, Csányi G. Hyperactive learning for data-driven interatomic potentials. *Npj Comput Mater* (2023) 9:168. doi:10.1038/s41524-023-01104-6

145. Podryabinkin EV, Shapeev AV. Active learning of linearly parametrized interatomic potentials. *Comput Mater Sci* (2017) 140:171–80. doi:10.1016/j.commatsci.2017.08.031

146. Li Z, Kermode JR, De Vita A. Molecular dynamics with on-the-fly machine learning of quantum-mechanical forces. *Phys Rev Lett* (2015) 114:096405. doi:10.1103/PhysRevLett.114.096405

147. Vandermause J, Torrisi SB, Batzner S, Xie Y, Sun L, Kolpak AM, et al. On-the-fly active learning of interpretable Bayesian force fields for atomistic rare events. *Npj Comput Mater* (2020) 6:20. doi:10.1038/s41524-020-0283-z

148. Zhang Y, Wang H, Chen W, Zeng J, Zhang L, Wang H, et al. DP-GEN: a concurrent learning platform for the generation of reliable deep learning based potential energy models. *Computer Phys Commun* (2020) 253:107206. doi:10.1016/j.cpc.2020.107206

149. Noid WG, Szukalo RJ, Kidder KM, Lesniewski MC. Rigorous progress in coarse-graining. *Annu Rev Phys Chem* (2024) 75:21–45. doi:10.1146/annurev-physchem-062123-010821

150. Guenza MG. Everything you want to know about coarse-graining and never dared to ask: macromolecules as a key example. *WIREs Comput Mol Sci* (2025) 15:e70022. doi:10.1002/wcms.70022

151. Li Z, Wellawatte GP, Chakraborty M, Gandhi HA, Xu C, White AD. Graph neural network based coarse-grained mapping prediction. *Chem Sci* (2020) 11:9524–31. doi:10.1039/D0SC02458A

152. Shinkle E, Pachalieva A, Bahl R, Matin S, Gifford B, Craven GT, et al. Thermodynamic transferability in coarse-grained force fields using graph neural networks. *J Chem Theor Comput* (2024) 20:10524–39. doi:10.1021/acs.jctc.4c00788

153. Yu Y, Harlim J, Huang D, Li Y. Learning coarse-grained dynamics on graph. *Physica D: Nonlinear Phenomena* (2025) 481:134801. doi:10.1016/j.physd.2025.134801

154. Wilson MO, Huang DM. Anisotropic molecular coarse-graining by force and torque matching with neural networks. *The J Chem Phys* (2023) 159:024110. doi:10.1063/5.0143724

155. Dhamankar S, Webb MA. Chemically specific coarse-graining of polymers: methods and prospects. *J Polym Sci* (2021) 59:2613–43. doi:10.1002/pol.20210555

156. Waltmann C, Wang Y, Yang C, Kim S, Voth GA. MSBack: multiscale backmapping of highly coarse-grained proteins using constrained diffusion. *J Chem Theor Comput* (2025) 21:6184–93. doi:10.1021/acs.jctc.5c00459

157. Kementzidis G, Wong E, Nicholson J, Xu R, Deng Y. An iterative framework for generative backmapping of coarse grained proteins. *arXiv E-Prints* (2025). doi:10.48550/arXiv.2505.18082

158. Ge Y, Zhu Q, Wang X, Ma J. Coarse-grained models for ionic liquids and applications to biological and electrochemical systems. *Ind Chem Mater* (2025) 3:383–411. doi:10.1039/D5IM00021A

159. Gerakinis D-P, Ricci E, Giannakopoulos G, Karkaletsis V, Theodorou DN, Vergadou N. Molecular simulation of coarse-grained systems using machine learning. In: *Proceedings of the 13th Hellenic conference on artificial intelligence SETN '24*. New York, NY, USA: Association for Computing Machinery. doi:10.1145/3688671.3688739

160. Han M, Sun G, de Pablo JJ. Attention-based functional-group coarse-graining: a deep learning framework for molecular prediction and design. arXiv:2502 (2025). doi:10.48550/arXiv.2502.00910

161. Ye H, Xian W, Li Y. Machine learning of coarse-grained models for organic molecules and polymers: progress, opportunities, and challenges. *ACS Omega* (2021) 6:1758–72. doi:10.1021/acsomega.0c05321

162. Greener JG, Jones DT. Differentiable molecular simulation can learn all the parameters in a coarse-grained force field for proteins. *PLOS ONE* (2021) 16:e0256990. doi:10.1371/journal.pone.0256990

163. Carrer M, Cezar HM, Bore SL, Ledum M, Cascella M. Learning force field parameters from differentiable particle-field molecular dynamics. *J Chem Inf Model* (2024) 64:5510–20. doi:10.1021/acs.jcim.4c00564

164. Christiansen H, Maruyama T, Errica F, Zaverkin V, Takamoto M, Alesiani F. Fast, modular, and differentiable framework for machine learning-enhanced molecular simulations. arXiv e-prints (2025). doi:10.48550/arXiv.2503.20541

165. Thaler S, Zavadlav J. Learning neural network potentials from experimental data *via* differentiable trajectory Reweighting. *Nat Commun* (2021) 12:6884. doi:10.1038/s41467-021-27241-4

166. Wang C, Pérez de Alba Ortíz A, Dijkstra M. Inverse design method with enhanced sampling for complex open crystals: application to novel zeolite self-assembly. *ACS Nano* (2025) 19:17423–37. doi:10.1021/acsnano.4c17597

167. Ashraf AB, Rao CS. Multiobjective temperature trajectory optimization for unseeded batch cooling crystallization of aspirin. *Comput and Chem Eng* (2022) 160:107704. doi:10.1016/j.compchemeng.2022.107704

168. Hensley A, Videbæk TE, Seyforth H, Jacobs WM, Rogers WB. Macroscopic photonic single crystals *via* seeded growth of DNA-coated colloids. *Nat Commun* (2023) 14:4237. doi:10.1038/s41467-023-39992-3

169. Liu H, Matthies M, Russo J, Rovigatti L, Narayanan RP, Diep T, et al. Inverse design of a pyrochlore lattice of DNA origami through model-driven experiments. *Science* (2024) 384:776–81. doi:10.1126/science.adl5549

170. Ramesh PS, Patra TK. Polymer sequence design *via* active learning. arXiv e-prints (2021). doi:10.48550/arXiv.2111.09659

171. Spyriouni T, Tzoumanekas C, Theodorou D, Müller-Plathe F, Milano G. Coarse-Grained and reverse-mapped united-atom simulations of long-chain atactic polystyrene melts: structure, thermodynamic properties, chain conformation, and entanglements. *Macromolecules* (2007) 40:3876–85. doi:10.1021/ma0700983

172. Sigalas NI, Anogiannakis SD, Theodorou DN, Lyulin AV. A coarse-grained model for capturing the helical behavior of isotactic polypropylene. *Soft Matter* (2022) 18:3076–86. doi:10.1039/D2SM00200K

173. Baig C, Mavrantzas VG. Multiscale simulation of polymer melt viscoelasticity: expanded-Ensemble monte carlo coupled with atomistic nonequilibrium molecular dynamics. *Phys Rev B* (2009) 79:144302. doi:10.1103/PhysRevB.79.144302

174. Shireen Z, Weeratunge H, Menzel A, Phillips AW, Larson RG, Smith-Miles K, et al. A machine learning enabled hybrid optimization framework for efficient coarse-graining of a model polymer. *Npj Comput Mathematics* (2022) 8:224. doi:10.1038/s41524-022-00914-4

175. Raissi M, Perdikaris P, Karniadakis GE. Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *J Comput Phys* (2019) 378:686–707. doi:10.1016/j.jcp.2018.10.045

176. Brunton SL, Proctor JL, Kutz JN. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proc Natl Acad Sci* (2016) 113:3932–7. doi:10.1073/pnas.1517384113

177. Schmidt M, Lipson H. Distilling free-form natural laws from experimental data. *Science* (2009) 324:81–5. doi:10.1126/science.1165893

178. Li Z, Kovachki N, Azizzadenesheli K, Liu B, Bhattacharya K, Stuart A, et al. Fourier neural operator for parametric partial differential equations. arXiv e-prints (2020). doi:10.48550/arXiv.2010.08895

179. Lu L, Jin P, Pang G, Zhang Z, Karniadakis GE. Learning nonlinear operators *via* DeepONet based on the universal approximation theorem of operators. *Nat Machine Intelligence* (2021) 3:218–29. doi:10.1038/s42256-021-00302-5

180. Cai S, Wang Z, Lu L, Zaki TA, Karniadakis GE. DeepM&Mnet: inferring the electroconvection multiphysics fields based on operator approximation by neural networks. *J Comput Phys* (2021) 436:110296. doi:10.1016/j.jcp.2021.110296

181. Lin C, Li Z, Lu L, Cai S, Maxey M, Karniadakis GE. Operator learning for predicting multiscale bubble growth dynamics. *The J Chem Phys* (2021) 154:104118. doi:10.1063/5.0041203

182. Lütjens B, Crawford CH, Watson CD, Hill C, Newman D. Multiscale neural operator: learning fast and grid-independent PDE solvers (2022). doi:10.48550/arXiv.2207.11417

183. Sammüller F, Robitschko S, Hermann S, Schmidt M. Hyperdensity functional theory of soft matter. *Phys Rev Lett* (2024) 133:098201. doi:10.1103/PhysRevLett.133.098201

184. Dijkman J, Dijkstra M, van Roij R, Welling M, van de Meent J-W, Ensing B. Learning neural free-energy functionals with pair-correlation matching. *Phys Rev Lett* (2025) 134:056103. doi:10.1103/PhysRevLett.134.056103

185. Robitschko S, Sammüller F, Schmidt M, Hermann S. Hyperforce balance *via* thermal noether invariance of any observable. *Commun Phys* (2024) 7:103. doi:10.1038/s42005-024-01568-y

186. Kampa SM, Sammüller F, Schmidt M, Evans R. Metadensity functional theory for classical fluids: extracting the pair potential. *Phys Rev Lett* (2025) 134:107301. doi:10.1103/PhysRevLett.134.107301

187. Zhu M, Zhang H, Jiao A, Karniadakis GE, Lu L. Reliable extrapolation of deep neural operators informed by physics or sparse observations. *Computer Methods Appl Mech Eng* (2023) 412:116064. doi:10.1016/j.cma.2023.116064

188. Maust H, Li Z, Wang Y, Leibovici D, Bruno O, Hou T, et al. Fourier continuation for exact derivative computation in physics-informed neural operators. (2022) doi:10.48550/arXiv.2211.15960

189. Brockherde F, Vogt L, Li L, Tuckerman ME, Burke K, Müller K-R. Bypassing the kohn-sham equations with machine learning. *Nat Commun* (2017) 8:872. doi:10.1038/s41467-017-00839-3

190. Snyder JC, Rupp M, Hansen K, Müller K-R, Burke K. Finding density functionals with machine learning. *Phys Rev Lett* (2012) 108:253002. doi:10.1103/PhysRevLett.108.253002

191. Li L, Snyder JC, Pelaschier IM, Huang J, Niranjan U, Duncan P, et al. Understanding machine-learned density functionals. *Int J Quan Chem* (2016) 116:819–33. doi:10.1002/qua.25040

192. Kasim MF, Vinko SM. Learning the exchange-correlation functional from nature with fully differentiable density functional theory. *Phys Rev Lett* (2021) 127:126403. doi:10.1103/PhysRevLett.127.126403

193. Nagai R, Akashi R, Sugino O. Machine-learning-based exchange correlation functional with physical asymptotic constraints. *Phys Rev Res* (2022) 4:013106. doi:10.1103/PhysRevResearch.4.013106

194. Li L, Hoyer S, Pederson R, Sun R, Cubuk ED, Riley P, et al. Kohn-sham equations as regularizer: building prior knowledge into machine-learned physics. *Phys Rev Lett* (2021) 126:036401. doi:10.1103/PhysRevLett.126.036401

195. Sripada S, Gaitonde AU, Weibel JA, Marconnet AM. Robust inverse parameter fitting of thermal properties from the laser-based Ångstrom method in the presence of measurement noise using physics-informed neural networks (PINNs). *J Appl Phys* (2024) 135:225106. doi:10.1063/5.0206247

196. Li Z, Huang DZ, Liu B, Anandkumar A. Fourier neural operator with learned deformations for PDEs on general geometries. *J Machine Learn Res* (2023) 24:1–26. doi:10.5555/3648699.3649087

197. Kaandorp MLA, Dwight RP. Data-driven modelling of the Reynolds stress tensor using random forests with invariance. *Comput and Fluids* (2020) 202:104497. doi:10.1016/j.compfluid.2020.104497

198. Wang J-X, Huang J, Duan L, Xiao H. Prediction of reynolds stresses in high-mach-number turbulent boundary layers using physics-informed machine learning. *Theor Comput Fluid Dyn* (2019) 33:1–19. doi:10.1007/s00162-018-0480-2

199. Liu W, Fang J, Rolfo S, Moulinec C, Emerson DR. An iterative machine-learning framework for RANS turbulence modeling. *Int J Heat Fluid Flow* (2021) 90:108822. doi:10.1016/j.ijheatfluidflow.2021.108822

200. Taghizadeh S, Witherden FD, Girimaji SS. Turbulence closure modeling with data-driven techniques: physical compatibility and consistency considerations. *New J Phys* (2020) 22:093023. doi:10.1088/1367-2630/abadb3

201. Matai R. Surrogate modeling for flow simulations using design variable-coded deep learning networks. *J Eng Appl Sci* (2025) 72:66. doi:10.1186/s44147-025-00634-8

202. Mukhtar A, Yasir ASHM, Nasir MFM. A machine learning-based comparative analysis of surrogate models for design optimisation in computational fluid dynamics. *Heliyon* (2023) 9:e18674. doi:10.1016/j.heliyon.2023.e18674

203. Smith AM, Katz D, Niemeyer K. Software citation principles. *PeerJ Computer Sci* (2016) 2:e86. doi:10.7717/peerj-cs.86

204. Councilman A, Fu D, Gupta A, Wang C, Grove D, Wang Y-X, et al. Towards formal verification of LLM-generated code from natural language prompts (2025). doi:10.48550/arXiv.2507.13290

205. Kalhor G, Ali S, Mashhadi A. Measuring biases in AI-generated co-authorship networks. *EPJ Data Sci* (2025) 14:38–3. doi:10.1140/epjds/s13688-025-00555-9

206. Klump J, Wyborn L, Wu M, Martin J, Downs RR, Asmi A. Versioning data is about more than revisions: a conceptual framework and proposed principles. *Data Sci J* (2021) 20:12. doi:10.5334/dsj-2021-012

207. Hinsen K. Dealing with software collapse. *Comput Sci and Eng* (2019) 21:104–8. doi:10.1109/MCSE.2019.2900945

208. Fursin G. Collective knowledge: organizing research projects as a database of reusable components and portable workflows with common interfaces. *Philosophical Trans R Soc A: Math Phys Eng Sci* (2021) 379:20200211. doi:10.1098/rsta.2020.0211

209. Drummond DC. Replicability is not reproducibility: nor is it good science (2025). Available online at: https://web-archive.southampton.ac.uk/cogprints.org/7691/(Accessed September 13, 2025).

210. Olson RS, La Cava W, Orzechowski P, Urbanowicz RJ, Moore JH. PMLB: a large benchmark suite for machine learning evaluation and comparison. *BioData Mining* (2017) 10:36. doi:10.1186/s13040-017-0154-4

211. Takamoto M, Praditia T, Leiteritz R, MacKinlay D, Alesiani F, Pflüger D, et al. PDEBENCH: an extensive benchmark for scientific. *Machine Learn* (2022). doi:10.48550/arXiv.2210.07182

212. Wang A, Hertzmann A, Russakovsky O. Benchmark suites instead of leaderboards for evaluating AI fairness. *Patterns* (2024) 5. doi:10.1016/j.patter.2024.101080

213. Spoczynski M, Melara MS, Szyller S. Atlas: a framework for ML lifecycle provenance and transparency (2025). doi:10.48550/arXiv.2502.19567

214. Han R, Zheng M, Byna S, Tang H, Dong B, Dai D, et al. PROV-IO+: a cross-platform provenance framework for scientific data on HPC systems (2023). doi:10.48550/arXiv.2308.00891

215. Namaki MH, Floratou A, Psallidas F, Krishnan S, Agrawal A, Wu Y, et al. Vamsa: automated provenance tracking in data science scripts (2020). 1542, 51. doi:10.1145/3394486.3403205

216. Gierend K, Krüger F, Genehr S, Hartmann F, Siegel F, Waltemath D, et al. Provenance information for biomedical data and workflows: scoping review. *J Med Internet Res* (2024) 26. doi:10.2196/51297

217. Ahmed M, Dar AR, Helfert M, Khan A, Kim J. Data provenance in healthcare: approaches, challenges, and future directions. *Sensors (Basel)* (2023) 23:6495. doi:10.3390/s23146495

218. Schelter S, Guha S, Grafberger S. Automated provenance-based screening of ML data preparation pipelines. *Datenbank-Spektrum* (2024) 24:187–96. doi:10.1007/s13222-024-00483-4

219. Pepe F, Zampetti F, Mastropaolo A, Bavota G, Di Penta M. A taxonomy of self-admitted technical debt. In: *Deep learning systems* (2024). doi:10.48550/arXiv.2409.11826

220. Ximenes R, Santos Alves AP, Escovedo T, Spinola R, Kalinowski M. Investigating issues that lead to code technical debt in machine learning systems (2025). 173, 83. doi:10.1109/cain66642.2025.00028

221. Wang J, Li L, Zeller A. Restoring execution environments of jupyter notebooks. In: *Proceedings of the 43rd international conference on software engineering ICSE '21*. Madrid, Spain: IEEE Press. p. 1622–33. doi:10.1109/ICSE43902.2021.00144

222. Saeed Siddik M, Li H, Bezemer C-P. A systematic literature review of software engineering research on jupyter notebook (2025). doi:10.48550/arXiv.2504.16180

223. Grayson S, Marinov D, Katz DS, Milewicz R. Automatic reproduction of workflows in the snakemake workflow catalog and nf-core registries. In: *Proceedings of the 2023 ACM conference on reproducibility and replicability ACM REP '23*. New York, NY, USA: Association for Computing Machinery. p. 74–84. doi:10.1145/3589806.3600037

224. Peer L, Biniossek C, Betz D, Christian T-M. Reproducible research publication workflow: a canonical workflow framework and FAIR digital object approach to quality research output. *Data Intelligence* (2022) 4:306–19. doi:10.1162/dint_a_00133

225. Lucaj L, Loosley A, Jonsson H, Gasser U, van der Smagt P. TechOps: technical documentation templates for the AI Act (2025). 8, 1647, 60. doi:10.1609/aies.v8i2.36663

226. Golpayegani D, Hupont I, Panigutti C, Pandit HJ, Schade S, O'Sullivan D, et al. AI cards: towards an applied framework for machine-readable AI and risk documentation inspired by the EU AI Act (2024). 48, 72. doi:10.1007/978-3-031-68024-3_3

227. Brajovic D, Renner N, Goebels VP, Wagner P, Fresz B, Biller M, et al. Model reporting for certifiable AI: a proposal from merging EU regulation into AI development (2023). doi:10.48550/arXiv.2307.11525

228. Bogucka E, Constantinides M, Šćepanović S, Quercia D. Co-designing an AI impact assessment report template with AI practitioners and AI compliance experts (2024). 7, 168, 80. doi:10.1609/aies.v7i1.31627

229. Author Anonymous High-level summary of the AI Act EU artificial intelligence act. Available online at: https://artificialintelligenceact.eu/high-level-summary/ [Accessed September 14, 2025]

230. Chirigati F, Rampin R, Shasha D, Freire J. ReproZip: computational reproducibility with ease. In: *Proceedings of the 2016 international conference on management of data SIGMOD '16*. New York, NY, USA: Association for Computing Machinery. p. 2085–8. doi:10.1145/2882903.2899401

231. Chirigati F, Shasha D, Freire J. ReproZip: using provenance to support computational reproducibility. In: *Proceedings of the 5th USENIX workshop on the theory and practice of provenance TaPP '13*. USA: USENIX Association. p. 1–4.

232. Steeves V, Rampin R, Chirigati F. Reproducibility, preservation, and access to research with ReproZip and ReproServer. *IQ* (2020) 44:1–11. doi:10.29173/iq969

233. Chirigati F, Shasha D, Freire J. Packing experiments for sharing and publication. In: *Proceedings of the 2013 ACM SIGMOD international conference on management of data SIGMOD '13*. New York, NY, USA: Association for Computing Machinery. p. 977–80. doi:10.1145/2463676.2465269

234. Lasser J. Creating an executable paper is a journey through open science. *Commun Phys* (2020) 3:143. doi:10.1038/s42005-020-00403-4

235. Kluyver T, Ragan-Kelley B, Pérez F, Granger B, Bussonnier M, Frederic J, et al. *Jupyter Notebooks—A publishing format for reproducible computational workflows*. Amsterdam, Netherlands: IOS Press. 87–90. doi:10.3233/978-1-61499-649-1-87

236. Perkel JM. Why jupyter is data scientists' computational notebook of choice. *Nature* (2018) 563:145–6. doi:10.1038/d41586-018-07196-1

237. Samuel S, Mietchen D. FAIR jupyter: a knowledge graph approach to semantic sharing and granular exploration of a computational notebook reproducibility dataset. *Trans Graph Data Knowledge* (2024) 2(4):1–4. doi:10.4230/TGDK.2.2.4

238. Pimentel JF, Murta L, Braganholo V, Freire J. Understanding and improving the quality and reproducibility of jupyter notebooks. *Empirical Softw Eng* (2021) 26:65. doi:10.1007/s10664-021-09961-9

239. Rule A, Birmingham A, Zuniga C, Altintas I, Huang S-C, Knight R, et al. Ten simple rules for writing and sharing computational analyses in jupyter notebooks. *PLOS Comput Biol* (2019) 15:e1007007. doi:10.1371/journal.pcbi.1007007

240. Masera M, Leone A, Köster J, Molineris I. Snakemaker: seamlessly transforming ad-hoc analyses into sustainable snakemake workflows with generative AI (2025). doi:10.48550/arXiv.2505.02841

241. Di Tommaso P, Chatzou M, Floden EW, Barja PP, Palumbo E, Notredame C. Nextflow enables reproducible computational workflows. *Nat Biotechnol* (2017) 35:316–9. doi:10.1038/nbt.3820

242. Djaffardjy M, Marchment G, Sebe C, Blanchet R, Bellajhame K, Gaignard A, et al. Developing and reusing bioinformatics data analysis pipelines using scientific workflow systems. *Comput Struct Biotechnol J* (2023) 21:2075–85. doi:10.1016/j.csbj.2023.03.003

243. Mölder F, Jablonski KP, Letcher B, Hall MB, Tomkins-Tinch CH, Sochat V, et al. Sustainable data analysis with snakemake. *F1000Res* (2021) 10:33. doi:10.12688/f1000research.29032.2

244. Jacobsen A, de Miranda Azevedo R, Juty N, Batista D, Coles S, Cornet R, et al. FAIR principles: interpretations and implementation considerations. *Data Intelligence* (2020) 2:10–29. doi:10.1162/dint_r_00024

245. Nunes JL, Barbosa GDJ, de Souza CS, Lopes H, Barbosa SDJ. Using model cards for ethical reflection: a qualitative exploration. In: *Proceedings of the 21st Brazilian symposium on human factors in computing systems IHC '22*. New York, NY, USA: Association for Computing Machinery. p. 1–11. doi:10.1145/3554364.3559117

246. Gebru T, Morgenstern J, Vecchione B, Vaughan JW, Wallach H, III HD, et al. Datasheets for datasets. *Commun ACM* (2021). 64:86–92. doi:10.1145/3458723

247. Alkemade H, Claeyssens S, Colavizza G, Freire N, Lehmann J, Neudecker C, et al. Datasheets for digital cultural heritage datasets. *J Open Humanit Data* (2023). 9:17. doi:10.5334/johd.124

248. Zargari Marandi R, Svane Frahm A, Milojevic M. Datasheets for AI and medical datasets (DAIMS): a data validation and documentation framework before machine learning analysis in medical research (2025). doi:10.48550/arXiv.2501.14094

249. Köster J, Rahmann S. Snakemake—A scalable bioinformatics workflow engine. *Bioinformatics* (2012) 28:2520–2. doi:10.1093/bioinformatics/bts480

250. De Nies T, Coppens S, Mannens E, Van de Walle R. Modeling uncertain provenance and provenance of uncertainty in W3C PROV. In: *Proceedings of the 22nd international conference on world wide web WWW '13 companion*. New York, NY, USA: Association for Computing Machinery. p. 167–8. doi:10.1145/2487788.2487871

251. Steidl M, Felderer M, Ramler R. The pipeline for the continuous development of artificial intelligence models—Current state of research and practice. *J Syst Softw* (2023) 199:111615. doi:10.1016/j.jss.2023.111615

252. MLOps. Continuous delivery and automation pipelines in machine learning | cloud architecture center. *Google Cloud*. Available online at: https://cloud.google.com/architecture/mlops-continuous-delivery-and-automation-pipelines-in-machine-learning (Accessed September 13, 2025).

253. Sustainable engineering of machine learning-enabled systems: a systematic mapping study. *Res Square*. doi:10.21203/rs.3.rs-4694122/v1

254. Bhatt H, Biswas S, Rakhunathan S, Vaidhyanathan K. HarmonE: a self-adaptive approach to architecting sustainable MLOps (2025). doi:10.48550/arXiv.2505.13693

255. Lanubile F, Martínez-Fernández S, Quaranta L. Teaching MLOps in higher education through project-based learning. In: *2023 IEEE/ACM 45th international conference on software engineering: software engineering education and training (ICSE-SEET)*. p. 95–100. doi:10.1109/ICSE-SEET58685.2023.00015

256. Eken B, Pallewatta S, Tran NK, Tosun A, Babar MA. A multivocal review of MLOps practices, challenges and open issues. *ACM Comput Surv* (2025) 58:1–35. doi:10.1145/3747346

257. Yakutovich AV, Eimre K, Schütt O, Talirz L, Adorf CS, Andersen CW, et al. AiiDAlab – an ecosystem for developing, executing, and sharing scientific workflows. *Comput Mater Sci* (2021) 188:110165. doi:10.1016/j.commatsci.2020.110165

258. Hymel S, Banbury C, Situnayake D, Elium A, Ward C, Kelcey M, et al. Edge impulse: an MLOps platform for tiny machine learning (2023). doi:10.48550/arXiv.2212.03332

259. Reichert P, Ma K, Höge M, Fenicia F, Baity-Jesi M, Feng D, et al. Metamorphic testing of machine learning and conceptual hydrologic models. *Hydrol Earth Syst Sci* (2024) 28:2505–29. doi:10.5194/hess-28-2505-2024

260. Liang W, Rajani N, Yang X, Ozoani E, Wu E, Chen Y, et al. Systematic analysis of 32,111 AI model cards characterizes documentation practice in AI. *Nat Machine Intelligence* (2024) 6:744–53. doi:10.1038/s42256-024-00857-z

261. Amith MT, Cui L, Zhi D, Roberts K, Jiang X, Li F, et al. Toward a standard formal semantic representation of the model card report. *BMC Bioinformatics* (2022) 23:281. doi:10.1186/s12859-022-04797-6