Intelligent Modeling and Simulation Life Cycle

Philipp Andelfinger
Nanyang Technological University, Singapore
philipp.andelfinger@ntu.edu.sg

Alessandro Pellegrini
Tor Vergata University of Rome, Italy
a.pellegrini@ing.uniroma2.it

Christopher D. Carothers
Rensselaer Polytechnic Institute, Troy, NY, USA
carotc@rpi.edu

Margaret Loper
Georgia Institute of Technology, Atlanta, GA, USA
Margaret.Loper@gtri.gatech.edu

Wen Jun Tan
Nanyang Technological University, Singapore
wjtan@ntu.edu.sg

Verena Wolf
German Research Center for Artificial Intelligence (DFKI), Saarbrücken, Germany
verena.wolf@dfki.de

Wentong Cai
Nanyang Technological University, Singapore
aswtcai@ntu.edu.sg

Abstract

Modeling and simulation is a well-proven approach for conducting what-if analyses of complex scenarios. However, the current societal and technical challenges require increasingly complex models and larger simulation experiments, which calls for more intelligent approaches to simulation and modeling in all aspects of simulation studies. To this end, we believe simulation and modeling can benefit from the recent advancements in machine learning and artificial intelligence (AI) as well as the emerging powerful and pervasive hardware and computing paradigms and systems. In this article, we examine the existing AI techniques and emerging hardware platforms in the context of the modeling and simulation life cycle, broadly comprising the stages of model creation, calibration, and experimentation. We identify key challenges on the path to deeper integration between AI and simulation techniques and outline research directions towards the vision of a higher degree of automation in simulation-supported scientific discovery.

1 Introduction

The quick advancement of machine learning and artificial intelligence (subsumed under the acronym AI for the remainder of the article) has significantly impacted various fields, including

Preprint – Submitted to ACM TOMACS. This version has not been peer-reviewed.

modeling and simulation (M&S). Modeling and simulation have long been fundamental tools for conducting what-if analyses of complex scenarios, facilitating decision-making in diverse domains such as engineering, healthcare, and environmental science. However, the escalating complexity of societal and technical challenges necessitates the development of increasingly sophisticated models and large-scale simulation experiments. This complexity, coupled with the limitations of traditional methods, underscores the urgent need for more intelligent approaches to M&S.

Recent breakthroughs in AI [165] offer exciting opportunities to enhance the M&S life cycle, encompassing model creation, calibration, and experimentation. These technologies, in conjunction with emerging hardware and computing paradigms, present unprecedented opportunities to improve the speed, accuracy, and capabilities of simulation studies. AI techniques can provide robust frameworks for addressing the intricate challenges associated with modern M&S. By leveraging these advanced techniques, it is possible to develop richer and more efficient simulation models that can adapt to the dynamic requirements of various applications.

The M&S life cycle traditionally involves several stages [168]: input modeling, concept model and validation, computational model and verification, experiment design, model execution, and output analysis. Each stage presents unique challenges and demands that can be significantly alleviated by integrating AI methodologies. For instance, process mining and symbolic regression can automate the creation of initial models, while AI-techniques for simulation-based inference and data assimilation can enhance a model's accuracy. During the execution phase, surrogate models can reduce computational costs, and deep learning-guided sampling strategies can accelerate simulation experiments.

The advent of new hardware technologies tailored for AI applications, such as neuromorphic devices and neural processing units, further amplifies the potential of intelligent M&S. These hardware advancements enable the efficient execution of complex models and support the deployment of AI-driven simulation frameworks on diverse computing platforms, including high-performance computing clusters and cloud environments. Already, the synergistic use of heterogeneous architectures, encompassing CPUs, GPUs, FPGAs, and accelerators alike allows for optimized performance and resource utilization, making it feasible to tackle large-scale simulations with higher fidelity and speed.

To fully harness these capabilities, the M&S community must adopt a unified approach that integrates AI techniques across the entire simulation life cycle. This integration will not only improve the accuracy and efficiency of simulations, but also facilitate the development of models that are better equipped to adapt to real-world dynamics and scenario scales. By understanding modern scientific studies as integrated simulation/AI endeavors, the benefits of simulation models based on well-understood mechanisms and of data-driven AI models are combined.

This paper highlights the requirements at each stage of the M&S life cycle in order to identify the major challenges and research directions on the road towards an intelligent modeling and simulation life cycle. In contrast to previous work at the intersection of simulation and AI [165], we clearly position the techniques, challenges, and research directions within the M&S life cycle. In addition, we explicitly account for the evolving, now largely AI-driven, hardware landscape and discuss its current uses in simulation studies as well as the research needs to make best use of increasingly heterogeneous platforms in the future. In this way, we aim to provide a cohesive and comprehensive framework that can enhance the capabilities and applications of M&S in various scientific and industrial domains. The article builds on discussions at the Dagstuhl seminar 22401, "Computer Science Methods for Effective and Sustainable Simulation Studies" [44, 283].

The remainder of this paper is structured as follows. Section 2 introduces the M&S life cycle, highlighting challenges and demands at the M&S stages using real-world applications as examples. In Section 3, we review notable AI techniques applied to M&S, categorizing them by life cycle stages and tasks to identify gaps requiring future research. We catalog modern and emerging computing systems, platforms, and hardware, along with new computational

paradigms that can support the M&S mission in Section 4. Based on all this, in Section 5 we identify areas that require further exploration and outline a roadmap for applying AI and emerging hardware platforms to the M&S life cycle. This roadmap is aimed at helping the research community in pinpointing potential opportunities for impactful contributions, and at fostering collaboration among the M&S and AI fields.

2 Modeling and Simulation Life Cycle and Research Architecture

This article provides a structured overview of how simulation studies can benefit from recent and emerging AI advancements and aims to identify current research gaps within the simulation life cycle. We take a two-pronged approach in organizing the material. First, to offer an accessible overview, the classical M&S life cycle is divided into three broad stages: model creation, model calibration, and simulation experiments. Second, we position the new advancements and resulting capabilities in a more detailed M&S research architecture comprised of the artifacts and processes involved in a M&S study. Below, we discuss several existing conceptualizations of simulation studies, based on which we design our coarse-grained and fine-grained organizations.

Among the most widely embraced M&S life cycles is the one proposed by Balci [27], which covers in detail the entities, processes, and responsibilities involved in a simulation study. The life cycle spans the stages from deriving a problem formulation from the "universe of discourse" to the presentation of the final results. The series of processes, some of which may be revisited as a study evolves, includes problem formulation, requirements engineering, conceptual modeling, architecting, design, implementation, integration, experimentation, and presentation. Before engaging in each of the processes, verification and validation efforts ensure that the created specifications, models, and data accord with previous specifications and the real-world problem to be addressed. Importantly, there is a clear distinction between the *conceptual model* and its implementation in the form of an *executable model* that is exercised as part of simulation experiments. A conceptual model is a living document that evolves from an informal to a formal description and serves as a means of communication among the participants in the simulation's development. It describes what is to be represented, the assumptions limiting those representations, and other capabilities (e.g., data) needed to satisfy the user's requirements. While an informal conceptual model may be written using natural language, a formal conceptual model is an unambiguous description of model structure in the form of mathematical and logical relationships among the system's components. The executable model is created by translating the conceptual model to a modeling language or general-purpose programming language, from which it is then compiled or interpreted for execution on a computer.

Banks et al. [29] offer a slightly more condensed overview of the steps in a simulation study that emphasizes specific needs of M&S studies. While their organization puts less emphasis on the need for verification and validation throughout the entire study and the possibility of having to revisit previous steps, it allows for a clearer delineation of broader multi-step stages. In Figure 1, we augment Banks et al.'s organization to highlight model creation, calibration, and simulation experiments as stages that are encountered in most typical simulation studies, and that each come with their own significant challenges. Based on this structure, in Section 3, we discuss model creation and model calibration separately before considering their unification and the production use of an executable model in simulation experiments.

An alternative view of simulation studies is given in the simulation research architecture by Ihrig et al. [136]. The focus of this architecture is on the relations among the target system to be simulated and the artifacts involved or created throughout the study. Different from the life cycles discussed above, knowledge in the form of existing domain theory is covered explicitly as a source to draw on in a model's creation. For the purposes of our survey, this representation allows us to pinpoint the processes in simulation studies that can benefit from specific AI-supported methods. To highlight the central role of the conceptual model and to make it easier

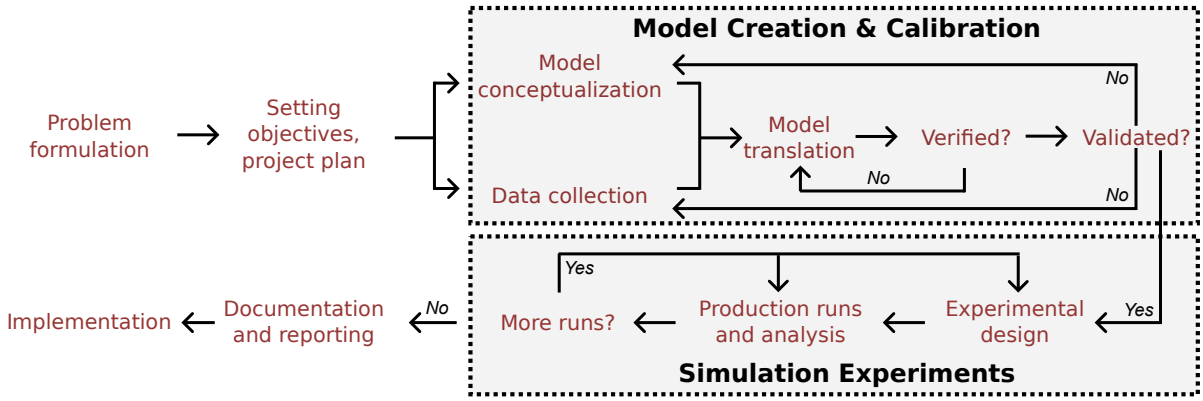

Figure 1: Steps in a simulation study, adapted from Banks et al. [29]. The broad stages of model creation, model calibration, and simulation experiments serve as a coarse-grained structure in Section 3.

to identify commonly encountered sequences of processes, we adapted and rearranged Ihrig's architecture to arrive at Figure 2. Here, the broader stages of model creation, calibration, and simulation experiments are reflected as sub-graphs that may be comprised of several artifacts and processes. For instance, model calibration involves generating simulated data in simulation experiments using the executable model and the comparison of the simulated data to empirical data. The numerical annotations are used in Table 1 to map entities and processes to concrete tasks that benefit from emerging AI approaches and techniques.

3 Emerging AI-Assisted Approaches for Modeling and Simulation

In this section, we walk through the main stages of the simulation life cycle, discussing recent and ongoing work that employs AI techniques that augment, simplify, or accelerate traditional manual, statistical, or algorithmic solutions to the problems addressed over the course of a simulation study. An overview of the roles that AI methods play in these stages and the employed techniques is given in Table 1.

3.1 Model Creation

On a high level [280], the creation of a simulation model involves forming hypotheses on causal relationships based on prior observations of some system and expressing the hypotheses in the form of a computer program that evolves a modeled system state over time. A more process-driven view describes model creation as a software engineering endeavor in which stakeholders, decision-makers, and model developers interact closely throughout a series of stages from an initial problem formulation to a final certified model [27]. Both perspectives describe the model creation as a largely manual effort in which modelers express hypotheses and domain knowledge in a chosen formalism, typically combining largely static mechanisms with structural and numerical parameters to be configured based on data during a separate calibration stage. This creates a sharp contrast to the largely data-driven creation of models in machine learning and artificial intelligence, where the separation between mechanisms and parameters is less clear. In the following, we will first adhere to this strict delineation between the model creation and calibration stages before discussing the gray area introduced by recent methods in Section 3.3.

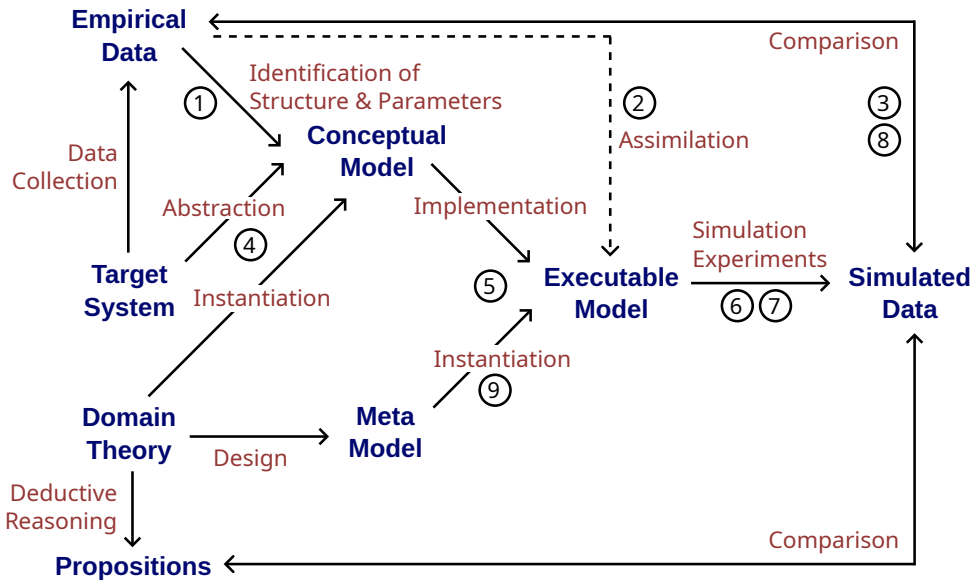

Figure 2: Simulation research architecture, adapted from [136] to emphasize the role of the conceptual model and the sequence of processes. The direct path from empirical data to the executable model (dashed arrow) is not part of traditional M&S life cycles, but represents the need for data assimilation in the context of Digital Twins. The circled numbers serve to map processes to the entries of Table 1.

3.1.1 Process Mining

Process mining (PM) techniques derive insights about business process executions from event data recorded by information systems [252]. Several types of PM exist, including *process discovery* (learning process models from event data), *conformance checking* (comparing event data with process models), and *process enhancement* (adding frequency or performance metrics to process models). For instance, discrete-event models of manufacturing systems can be extracted from event logs representing observations of the timing of order arrivals and processing steps on different equipment in the system [92]. Although process mining steps such as choosing suitable probability distributions can be automated, the underlying structure of possible target systems is still provided manually based on domain knowledge. Recently, large language models (LLMs) have emerged as conversational interfaces trained on extensive data [279], achieving near-human performance in various general tasks [328]. Their potential for PM lies in the embedded domain knowledge useful for generating database queries and insights [229], their logical and temporal reasoning capabilities [28, 176], and inference abilities over structured data [144]. LLMs facilitate the generation of textual descriptions from process data and the handling of inputs such as event logs or formal process models [34, 35]. They can also generate process models from textual descriptions, with recent work using LLMs to create models according to the business process model notation (BPMN) and declarative constraints from text [112].

3.1.2 Model Creation from Natural Language

An ambitious vision is to generate model formulations directly from human-provided descriptions in natural language. With the advent of transformer-based LLMs, code generation from natural language has become practical and found widespread use [145]. In a modeling and simulation context, reliable code generation in a domain-specific modeling language would empower non-technical domain experts to efficiently formalize their knowledge as simulation models. However, at present, specialized fine-tuning of the language models may be a prerequisite to ensure generalization of code generation models to less widely used languages [145]. Further limitations in the generalization capabilities have recently been demonstrated when attempting

to generate executable and concise simulation models in the three paradigms of discrete-event modeling, system dynamics, and agent-based modeling [93]. Ongoing efforts towards augmenting the decoding process to enforce adherence to a grammar or even the semantic correctness of the generated code [194, 83, 219] as well as to systematically infer the user's intent [236, 237] may ameliorate this situation. On the other hand, encouraging early results in generating simulation models from natural language have been reported in the problem domains of systems biology [185] and logistics [138]. At the time of writing, LLMs have already proven viable as assistants supporting the critical thinking required when forming hypotheses, creating a conceptual model, and during its formalization [2].

3.2 Model Calibration

The internal structure and state transition mechanisms defined in a simulation model are almost always configured by some numerical parameters. The calibration stage involves identifying values for the parameters that render the simulation model a sufficiently faithful approximation of the target system. One way to tackle calibration is through statistical point estimation, i.e., by finding a single parameter combination that satisfies some criterion. This approach typically takes the form of *simulation-based optimization*, in which simulation runs are alternated with invocations of a search strategy that proposes parameter combinations in order to minimize the deviation of the simulation output from the data. To combat the associated computational costs, approximate surrogate models are often used. Since the uses of simulation-based optimization and surrogate modeling extend far beyond calibration, we defer a more detailed discussion to Sections 3.4.

3.2.1 Simulation-based Inference

Bayesian methods offer a principled approach to inferring the statistical properties required for calibration [153]. In traditional Bayesian inference, modelers construct probabilistic models explicitly, e.g., in the form of graphical models that combine probability distributions assumed to define the target system's behavior. In many cases, this allows the desired statistical properties to be computed analytically based on the involved probability distributions.

Simulation models, however, typically do not offer this form of analytical tractability. Thus, in *simulation-based inference* [71] (SBI, also referred to as *likelihood-free inference*), the model behavior can only be characterized by sampling the simulation output at different points of the parameter space, creating challenges in terms of computational cost.

Importantly, SBI characterizes the probability distribution of the parameters' values given the data. Hence, the results describe the level of uncertainty in the parameter's values. For example, SBI has been applied to characterize the uncertainty in simulations for safety evaluations of self-driving systems [133] and in simulation models of cardiovascular systems [292]. The identified distributions provide a rigorous justification for a calibration either by sampling parameter values from the acquired distribution or by maximizing the parameters' likelihood.

At the heart of Bayesian inference is Bayes' theorem:

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)}. \tag{1}$$

In this context, A represents the model parameters and B are the data. Classical inference methods either maximize the likelihood $P(B \mid A)$ to obtain a maximum likelihood estimate (MLE) of the parameters given the data or maximize the posterior $P(A \mid B)$ to obtain a maximum a posteriori (MAP) estimate. Alternatively, they may sample from the posterior using Markov Chain Monte Carlo (MCMC) methods. These methods share the requirement that the right-hand side of Eq. 1, and in particular the likelihood function $P(B \mid A)$ can be evaluated. In SBI, the likelihood function must be approximated by sampling.

• **Approximate Bayesian Computation (ABC)**. A common approach for sampling-based estimation of the likelihood function is Approximate Bayesian Computation, which randomly samples model parameter combinations and accepts simulation outputs within some ϵ-region around the data. Although various improvements to the sampling efficiency of ABC have been proposed, its enormous computational demands typically necessitate a dimensionality reduction using summary statistics on the parameter space and/or concessions in terms of the approximation quality configured through ϵ. Neural networks can be used to automate the choice of an appropriate summary statistic and thus to reduce the cost of ABC [143]. A sound choice of the summary statistics to learn based on simulation input-output pairs is the expectation of the posterior $\mathbb{E}[A \mid B]$, which leads to the final ABC posterior having the same mean as the exact posterior for $\epsilon \to 0$. It has been shown that ABC can be avoided and exact MCMC-based inference can be achieved if the simulator is *differentiable*, which allows a modification of Hamiltonian Monte Carlo to be applied directly [109]. Combined with recent advancements in achieving differentiability for various forms of simulations (cf. Section 3.4), this capability extends the reach of SBI towards challenging simulations such as those dominated by discrete control flow (e.g., agents making discrete decisions based on their environment).

• **Markov Chain Monte Carlo (MCMC)**. Having access to an approximation of the posterior or likelihood, a point estimate of the best-fitting parameters can be determined by maximizing the likelihood. More commonly, the parameters' distribution is characterized by sampling from the posterior using well-established MCMC methods, which are increasingly benefiting from machine learning methods as well.

A key component of MCMC algorithms is the transition kernel, which yields the probability $P(X_{t+1} \mid X_t)$ of the next point X_{t+1} in the parameter space to be visited conditioned on the current point X_t. Central to this transition kernel is the proposal distribution, which generates candidate points to explore in the parameter space. The proposal distribution determines how new samples are proposed, shaping the exploration of the space and influencing the likelihood of the proposed samples being accepted or rejected. The choice of the proposal distribution is critical because it directly impacts the efficiency of the algorithm. Recent adaptive methods consider the design of proposal distributions and transition kernels as reinforcement learning (RL) tasks [64, 291, 287].

MCMC methods such as Metropolis-Hastings rely on an appropriate proposal distribution to achieve fast convergence to the posterior. A natural way to make use of the approximation capabilities of neural networks is to adaptively adjust the proposal distribution according to the gathered samples. In this approach, the proposal distribution is a gradually improving approximation of the posterior, e.g., generated by a Bayesian neural network [125].

One limitation of MCMC-based sampling is its sequential nature, which can make it enormously time-consuming considering the need to reach a stationary distribution and the cost of evaluating complex posteriors, or approximations thereof. If the posterior can be expressed as a product of subposteriors, parallelization can be achieved by sampling from the subposteriors concurrently and combining the samples to determine a sample from the overall posterior. Recently, the use of so-called real non-volume preserving transformations (real NVPs), which represent invertible neural transformation among density functions, has enabled parallelized MCMC with communication cost independent of the number of subposteriors [198].

• **Distribution Learning**. Beyond ABC, the issue of efficiently approximating the conditional probability distributions from Bayes' theorem based on samples of a simulation model's input-output relation has been addressed using methods for *distribution learning*. In one family of approaches, a proposal prior $\tilde{P}(A)$ is trained together with a posterior estimate $\tilde{P}(A \mid B)$ by iteratively sampling from the current prior and updating the two approximations according to input-output pairs generated by the simulation in a Bayesian regime [215, 183]. The distributions are learned by mixture density networks, i.e., feed-forward neural networks that generate the parameters of Gaussian mixtures.

Instead of approximating the posterior directly, distribution learning can also be applied to the likelihood function, which avoids biasing the posterior estimation by the choice of pro-

posal prior [216]. Here, the density estimation relies on *normalizing flows*, which, similarly to NVPs, model a target probability distribution by transforming an input distribution through a sequence of bijections. Through their invertibility, normalizing flows allow their likelihood function to be evaluated directly. Normalizing flows have also been combined with *importance sampling* using intermediate learned distributions as proposals to increase the sampling efficiency [235]. An alternative to normalizing flows as approximators of complex likelihood functions is given by exponential families parametrized by neural networks. Compared to ABC, these likelihood-free methods based on density estimation have proven to reduce the required number of simulations for inference by several orders of magnitude [216].

Finally, a recent work allows inference via both the posterior and the likelihood by training a transformer to learn the joint distribution of the model parameters and the data, both of which are represented as tokens [104]. Via an attention mask, dependencies among parameters and observations are enforced, allowing users to encode prior knowledge.

3.2.2 Data Assimilation (DA)

With the recent shift from static models towards Digital Twins that continuously track the evolution of real-world systems, there is a growing need for data assimilation (DA) to ensure that simulation models remain aligned with incoming data [128]. Traditional data-driven methods for detecting change points that indicate the need for recalibration are well-established [6], but recent work increasingly leverages machine learning to improve both detection and model updating [58, 42]. For instance, by alternating simulation-based predictions of changes in system behavior and neural predictions of the corresponding likelihood of incoming data, high-quality estimates of change points can be determined [312].

DA is often implemented using Sequential Monte Carlo (SMC) methods, where a set of particles—representing candidate system states—are sampled from a prior distribution and iteratively updated to approximate the posterior distribution of system states given the observations. Since the probability associated with many of the particles will typically tend to zero, resampling is often applied to maintain a set of particles with non-negligible likelihoods. The resampling can be efficiently realized by a simulator's capability to roll back the simulation state to a previous point in simulation time, which is widely used for speculative parallel simulations [96]. As demonstrated in road traffic simulations, by repeatedly advancing the simulation for one period and gathering the model state before rolling back, a new set of particles can be generated efficiently [301]. SMC has also been adapted to support discrete-event simulations and applied to road traffic simulations [129].

Although it has been pointed out that DA for agent-based simulation is still a nascent area [101], several works have proposed Bayesian methods to update agent-based models to accord with real-time data dynamically. Several of these works rely on SMC to assimilate data into models of passenger mobility in transportation scenarios and point out the scalability challenges in terms of the number of particles required to handle large numbers of agents [187, 278]. The granular nature of agent-based models has been exploited to achieve highly diverse sets of particles in SMC [289]. During resampling, instead of starting from existing model states as a whole, attributes of individual agents—here representing occupants in a smart building—are combined across particles to increase the state diversity. An alternative demonstrated using the classical predator-prey model maintains the correlation structure among agents by determining a bounding volume over the likelihood function [274]. Identifying a suitable bounding volume can be automated via abstract interpretation [70]. Finally, DA via variants of the Kalman filter is efficient even in complex pedestrian simulation scenarios and supports the integration of categorical variables [66, 269].

Deep learning has been applied to improve state estimation fidelity beyond traditional approaches [19]. By leveraging observations, the current model state, and the outputs of conventional DA as training data, a neural network can learn to perform DA autonomously. When the simulation continues using the DA steps generated by the neural network, a feedback loop

emerges, enabling the learned assimilation function to account for the cumulative effects of previous assimilation steps.

As incoming data is frequently incomplete or subject to measurement noise, error correction methods play an important role when assimilating new data points. Neural networks can support this process by learning error correction terms to vastly accelerate the convergence to a ground truth [87], with the best results achieved using an *intrusive* form that embeds the error correction terms directly in the model dynamics. Notably, this setting can allow the adaptation to incoming data to be confined purely to the error-correcting terms, obviating the need to recalibrate the mechanistic model.

Going beyond DA for updating the model state alone, works originating in process mining have proposed methods for dynamic data-driven updates to the model structure itself. In wafer fabrication simulations, an approach has been proposed that integrates process mining to generate Petri nets representing the production flow with decision trees capturing the logic of queuing and batching operations [273].

3.3 Joint Model Creation and Calibration

Methods for creating partial or full simulation models from data are increasingly blurring the line between model creation and calibration. By identifying interpretable models directly from data, these works are stepping stones towards a higher degree of automation in the overall process of scientific discovery.

3.3.1 Symbolic Regression

In symbolic regression [186], models are generated by forming expressions that best approximate the data from libraries of state variables and operators, in some cases enforcing constraints to ensure the interpretability of the generated expressions [171]. While established methods rely on genetic programming to form expression trees [325, 326], the problem has also been cast as a RL task in which a policy is learned to iteratively select symbols to add to an expression [16, 186].

Recent works are moving towards neural approaches. In a direct but somewhat rigid approach, neural network-based symbolic regression can be achieved by training a neural network with appropriately configured activation functions [255]. While sparsity can be encouraged by augmenting the objective with suitable penalty terms, the static network topology and numerical limitations on the set of applicable operators constrain the diversity of expressions that can be generated. In deep symbolic regression [228], RL is combined with a sampling of expressions from a distribution defined by a recurrent neural network.

Another neural approach involves extracting symbolic representations from generic neural networks trained on the data. This process effectively creates a symbolic surrogate model, which not only enhances interpretability but can also improve generalization by capturing fundamental relationships in a structured, equation-like form [72]. Like in other areas of artificial intelligence, pre-trained *transformers* are increasingly becoming the architecture of choice. In an end-to-end training setting, a transformer is trained to map a set of input-output pairs to a sequence of tokens representing the output expression's symbols [36, 150]. Numerical constants are either adjusted using a traditional optimization algorithm in a separate refinement step or predicted directly by the transformer. This type of approach has been shown to predict ordinary differential equation models of dynamical systems from a single trajectory [79]. Recent advancements integrate Monte Carlo tree search into the transformer's decoding process to integrate external knowledge as feedback on the prediction quality [264].

A limitation of neural symbolic regression approaches is the need for—typically domain-specific—pre-training. Recently, foundation models that represent general relationships between the symbolic and numerical domains have been proposed to reduce the need for specialized training [193].

3.3.2 Program Synthesis

Building a simulation model typically requires integrating well-established structural and mechanistic properties of the target system with aspects that must be modeled stochastically or inferred from data. Going beyond the mathematical expressions generated by symbolic regression, program synthesis [188, 115] addresses the problem of generating entire programs that best fit some specifications given by data or other constraints. An important qualitative difference to code generation using LLMs (cf. Section 3.1.2) is that the specification is formal, making the program's adherence directly testable. While many of the existing works in program synthesis are considering relatively short programs, use cases such as synthesizing agent behaviors in multi-agent systems [207] show the promise of the approach for filling "gaps" where certain elements of a simulation model can only be described in terms of partial properties or constraints. Such approaches complement the more human-driven model creation from natural language discussed in Section 3.1.

Program synthesis and deep learning come together in *neurosymbolic programming* (NP) [55, 271]. Residing between the extremes of purely neural network-based learning approaches and traditional human-led programming practices, NP allows neural and symbolic program elements to be learned jointly [271], similarly to approaches that combine differential equations with neural model elements [240]. A key enabler of this approach are *domain-specific modeling languages* in which modelers can express constraints on the programs to be learned. The joint learning and synthesis process calls for modeling languages made up of differentiable building blocks to allow for a gradient-based search across the high-dimensional space created by the neural network parameters and the possible combinations of symbolic program elements (cf. Section 3.4.3). When generating simulation models with the help of NP, user knowledge may be introduced in the form of known partial program structures and logical propositions to be satisfied [304], or physical constraints such as valid units [277]. In the context of physics simulations, such approaches are termed *physics-informed machine learning*, with the user-induced structure or constraints interpreted as inductive biases [151, 196, 199].

3.4 Simulation Experiments

Once a simulation model has been created, calibrated, and shown to be valid for the purposes of a study, it is exercised as part of simulation experiments, frequently in the form of exploratory parameter sweeps or iterative runs for optimizing simulation output statistics. In many cases, the high dimensionality of the model parameter space puts a full exploration out of reach. This situation is compounded by the long execution times of many simulation models and the need to conduct several runs per parameter combination to reach sufficient confidence levels for stochastic models. Already, various AI methods are employed to make better use of the available resources by fast-to-execute models as well as intelligent schemes for optimization, sampling, and output analysis.

3.4.1 Runtime Support

Simulations share with many other computational problems the opportunity to choose from a portfolio of available algorithms. The automation of this decision-making process, typically focusing on computational efficiency, is known as *automated algorithm selection* [154], the main challenge being to identify and detect the algorithm and problem characteristics that imply a good match. For instance, there are several algorithms for stochastic simulations of chemical reaction systems based on the Gillespie algorithm [103], all equivalent in the generated state trajectories but vastly different in their computational cost depending on the dynamic state of the system. In this domain, *reinforcement learning* has successfully been employed to select an algorithm at runtime [85]. Similarly, when model components are available at different fidelity levels, an automatic selection mechanism can enhance simulation efficiency while ensuring the required level of accuracy [51, 60].

The selection problem becomes more involved in heterogeneous hardware environments, where decisions must be made among sequential, data-parallel, or pipelined execution alternatives on different computing devices. Here, the main focus is on characterizing the computational properties of different hardware devices such as CPU, GPUs, and FPGAs on one hand, and model components on the other, either directly based on empirical measurements or by pre-trained neural performance models [298].

In the context of parallelized simulations, the simulation performance hinges on the choice in synchronization algorithms [96], but also on the degree to which the configuration of various tuneable algorithm parameters harmonizes with the evolving model dynamics. Several works have formulated the resulting control problems as Markov decision problems to be tackled by reinforcement learning. *Multi-armed bandit models* and *Q-learning* have been employed for dynamic load balancing among processors [197, 174], to dynamically restrict the aggressiveness of speculative execution [288], and to optimize the snapshotting of model states required for simulation rollbacks [222].

3.4.2 Surrogate Modeling

Going back at least to the 1990s, the universal approximation capabilities of neural networks are widely used to learn surrogate models, in some contexts also referred to as meta-models, of mechanistic simulation models [310, 31]. A common motivation is the potential for faster execution if a sufficiently small neural network can approximate the input-output relation encoded in the original model. The literature on neural surrogates has closely followed the advancements in neural network model architectures, with recent works making use of *long short-term memory* (LSTM) networks to capture temporal dependencies [67], *graph neural networks* to approximate graph-centric models [38], and *transformer architectures* for their higher parallelizability compared to recurrent neural networks [318].

- **Sample-efficient learning for surrogate creation**. In comparison to many other training tasks, the repeatability of simulations allows ample training data to be generated. At the same time, care must be taken that the training cost does not outweigh the subsequent reduction in execution times. Modern training techniques aim to minimize the training time and to balance fidelity and execution time. Classical forms of dimensionality reduction such as *principal component analysis* as well as neural approaches such as *autoencoders* can reduce the number of samples required [126].

A common theme in the recent literature is to apply *transfer learning* based on multi-resolution data. In these approaches, the surrogate creation time is reduced by first training on data from low-resolution simulations and gradually refining the surrogate by training on higher-resolution simulations [263, 284, 317, 170].

To counter the biases of simulation models when compared to a real-world target system, transfer learning can also be applied to refine a surrogate model originally trained on simulation data using empirical data [164, 134, 245]. Compared to a purely data-driven creation of predictive models, this approach integrates the domain knowledge encoded in the simulation model while still correcting for systematic biases.

Another class of recent methods aims to reduce training times by *active learning*, in which simulation runs are requested dynamically only for areas in the parameter space deemed the most informative [26, 205]. This goal is shared with more classical examples of *sequential design strategies* that aim to reduce the required number of samples for surrogate modeling or sensitivity analysis [73], for instance, by sampling techniques based on Voronoi tessellation [74].

- **Generalizable surrogates**. Some simulation tasks permit the learning of generalizable model behaviors that can be exercised across various concrete model instantiations. For instance, by training a neural network to predict short-term trajectories generated by classical car following and lane changing models for road traffic simulations, the result of a sequence of time-stepped updates of the simulation state can be closely approximated by a single neural "fast-forwarding" step independently of the specific road network topology [13].

Similar approaches have been applied to accelerate molecular dynamics (MD) simulations, where generalizable surrogates learn fundamental molecular interactions rather than mimicking isolated simulation outputs. Typically, in MD simulations mutual forces between pairs of molecules are computed at each time step, creating substantial computational load even when studying relatively localized effects. Instead of training for a specific molecular system, surrogate models can generalize across different force fields, chemical compositions, and thermodynamic conditions. Key use cases for MD lie in designing new materials and for drug discovery, which require simulations of large numbers of molecules over extended time spans. The use of fast, yet accurate neural surrogates has put previously intractable experiment scales into reach, particularly when making use of large high-performance computing clusters [181]. The time-stepped nature of MD can be retained by training neural surrogates that estimate the inter-molecule or inter-atomic potentials, which are then still applied in a series of step-wise state updates as in classical MD. Here, active learning plays the important role of maximizing the utility of the reference trajectories from computationally expensive high-fidelity MD simulations [322]. Particularly when considering long-range interactions, active learning helps tackle the vastly increased feature spaces and computational demands [17].

Important use cases include determining the hydrophobicity of molecules, i.e., the degree to which they are repelled by water, which is key in predicting the success of a new drug. By way of a three-dimensional convolutional neural network, the required energy computations have been accelerated by several orders of magnitude from about 35 days of CPU time to only about 7 minutes on a graphics processing unit, with errors comparable to those inherent to the MD calculations [116].

Software frameworks such as TorchMD [221] and DeePMD-kit [315] support the close integration between MD simulations, training loops, and the execution of hybrid simulation-based/neural models. Recent efforts to maximize efficiency on the largest available heterogeneous compute clusters enable detailed simulations of up to 10^8 atoms [241, 181].

3.4.3 Simulation-based Optimization

In simulation-based optimization (SBO) [49], which includes model calibration as a special case, a simulation is evaluated at a series of points in the input parameter space in order to minimize or maximize an output statistic. The computational demand of SBO can be enormous due to the combined cost of repeated simulations and the high dimensionality of the input space in many SBO problems.

- **Surrogate-Based Optimization**. While surrogate models are commonly used to reduce the number of required simulation evaluations, recent research on machine-learning-based surrogates has shifted towards the use of surrogate ensembles, where solutions are determined either by weighting candidate solutions from different surrogates or by selecting the best identified solution. A common way to form ensembles is to approximate the sampled simulation output using different types of models such as artificial neural networks, support vector machines, and radial basis functions (e.g., [247, 327]). Alternatively, ensembles of the same type of surrogate model can be generated from the same data based on perturbations of the simulation output [173] or by clustering of the parameter space to handle multi-modality [142].

The availability of one or more surrogate models allows for a dynamic selection of the most appropriate model at each iteration of the optimization loop. A natural criterion for selection is the expected impact of a simulation evaluation on optimization progress. A related concept, *model preemption*, has been explored outside of surrogate modeling [246]. The rationale behind model preemption is that if it can be predicted that a partially evaluated solution will not contribute meaningfully to the optimization, the simulation run can be terminated early, and the result discarded. In road traffic simulations, a method inspired by this approach was proposed in which simulation evaluations with low expected objective function value are dynamically deferred to a computationally inexpensive surrogate model [11]. A closely related approach performs weighted sampling across a set of surrogate models at each optimization iteration,

with the model weights based on each model's expected fidelity [282]. In a form of incremental learning, the surrogates are trained on the fly based on the observed simulation outputs, split into a training set and an evaluation set to determine the current model weights.

• **Robust Design Optimization**. In robust design optimization, the goal is to find model parameters that lead to a simulated system behavior that is robust to noise representing randomness and uncertainties such as those created by manufacturing variations. *Distributionally Robust Optimization* (DRO) [242] focuses on the worst case attainable under the unknown distributions when evaluating candidate solutions. More recently, *Bayesian Risk Optimization* [297] has been proposed as a framework to handle uncertainty in SBO. By allowing the stakeholder's risk tolerance to be specified in the form of a risk function, the approach generalizes beyond DRO's worst-case assumptions. This approach has been extended to handle cases as encountered in Digital Twin settings, where data arrives sequentially in a streaming fashion, and where the data depends on the chosen values of the decision variables [178].

Closely related to their use in sequential design for surrogate modeling (cf. Section 3.4.2), neural networks have been also applied to problems of this type. Here, the goal is to find model parameters that lead to a simulated system behavior that is robust to noise representing randomness and uncertainties such as those created by manufacturing variations [53]. Since establishing robustness results requires several model evaluations around each candidate parameter combination, adequate surrogates are important tools to reduce the often prohibitive cost of comprehensive robust design optimization. In this problem setting, the fidelity of the surrogate is particularly critical as shifts in the optima can lead to falsely classifying design candidates as robust. A critical review of various forms of surrogates [53] has found a classical model based on *Analysis of Variance* (ANOVA) [239] to be more reliably accurate compared to neural networks.

Another approach is to resort to the original simulation model when evaluating designs, while neural networks steer the choice of candidate parameters. In a recent work, two *Generative Adversarial Networks* (GANs) compete in a setup where one network aims to generate robustness-optimizing parameter combinations, whereas a second network generates noise patterns aiming to maximally disrupt the system behavior [89].

• **Reinforcement Learning.** Simulations are commonly used in model-free reinforcement learning (RL) as representations of real target environments. Among the key challenges in RL is the credit assignment problem, i.e., attributing increases in the reward to specific actions taken. Large recurrent neural networks (RNNs) are highly expressive models that can learn rich spatial and temporal representations of data. RNN-based *world models* can be used to extract useful representations of space and time from the environment to train the policy [117]. In this way, world models facilitate generalization and allow learning action-reward relationships from projected outcomes [118], thereby reducing the number of simulation required during the training process.

RL using molecular dynamics simulations has been applied to de novo molecular systems in the context of drug discovery and material design. For instance, RL was used for improved sampling of polymer chain conformations by influencing the Brownian forces among molecules and thus steering the simulation towards desired states [107]. Similar works have employed RL via policy gradient [78]. Finally, *Expert Iteration* [18], which combines tree search to propose concrete actions with a neural network to generalize across the entire state space, has been used to identify starting configurations of molecular dynamics simulations that lead to desired target states [320].

• **Differentiable Simulation**. Among the main enablers for the enormous strides made in machine learning in the past decades is the capability to efficiently calculate partial derivatives of loss functions defined over neural networks, which allows gradient-based optimization methods to steer the network parameters towards a local optimum. The well-known backpropagation algorithm [253] is a special case of automatic differentiation (AD) [111], which subsumes methods that carry along derivative information with the mathematical operations performed in computer programs written in general-purpose programming languages. After its popular-

ization in the 1980s, AD found widespread use in solving optimization problems in modeling and simulation studies [230], often for finite element methods [82, 68] and differential equation models [140, 46].

As an AD run computes derivatives only along a single control flow path of the program, stochastic simulation models with discrete control flow such as event-driven models are not always directly amenable to AD. The literature on infinitesimal perturbation analysis [123] describes the conditions under which averaging of AD-determined derivatives provides an unbiased estimation as well as manual transformations to enforce these conditions for certain models [106].

One way to achieve differentiability is to create a neural network-based surrogate model, at the cost of extensive training and a loss in fidelity. In the past years, a myriad of directly differentiable simulators have been presented targeting specific application domains such as physics [258, 91, 132], robotics [130, 127, 121], and biology [209, 4, 137, 110]. The models implemented in these simulators are designed to support high-fidelity gradient estimates via AD. Since causes for discontinuities in such models are well-known, appropriate continuous approximations with acceptable error bounds can be chosen [270]. In frameworks for differentiable agent-based simulations [7, 61, 8], models are formulated purely using differentiable building blocks, typically involving approximations of discontinuous functions and control flow elements.

Recently, more generic AD-based gradient estimators applicable to programs with discrete control flow have emerged. These estimators combine path-wise gradients determined via AD over sampled program trajectories with estimates of the effects of conditional branches. Stochastic AD [22] takes a parametric approach, applying a custom chain rule that accounts for the distribution of the branch conditions values. For each sample taken, in addition to the "primal" control flow path, the computation follows an additional path that reflects the operations as if an alternative path had been taken. While this approach requires the distribution of the branch condition to be known, its parameters can result from previous operations. The *DiscoGrad gradient oracle* [160, 7] is a non-parametric estimator applicable to generic imperative programs. Here, both the effects of conditional branches and the conditions' distributions are estimated over a series of Monte Carlo samples, allowing differentiation across programs branching on arbitrary functions of random variables. These approaches have been applied to simulation-based optimization problems across several thousands of parameters [160].

In a related approach, the gradient of the expectation is computed via MCMC, which involves handling the discontinuities introduced by the rejection step. By differentiating through the entire MCMC sampling procedure, parameters that affect the posterior distribution can be adjusted directly in the course of the sampling [23].

Beyond their uses in optimization and calibration, differentiable simulation also unlocks opportunities to differentiate through combined computational graphs made up of neural networks and entire simulation models [202, 108, 270, 8]. In this combination, neural networks can make direct use of the training signal provided by a simulation model, in contrast to most traditional reinforcement learning methods, which treat the simulation as a black box.

3.4.4 Output Analysis

Once a simulation model has been constructed and calibrated, stakeholders and decision-makers may be interested in various properties of the simulation output beyond predictions of the system behavior. Classical statistical approaches focus on statistics across simulation runs in the form of point estimates of expectations or higher moments with confidence intervals [166] or parametric fits to output distributions [59].

• **Data Farming and Explainable Artificial Intelligence (XAI)**. Data farming [124] is a method that carries out large numbers of simulation runs using high-performance computing to gather simulation outputs across high-dimensional parameter spaces. Different from the targeted parameter explorations in optimization or calibration, data farming is more open-ended, providing data that can support various potential analyses such as outlier detection, hypoth-

esis formation, or sensitivity analysis. Rooted in military use cases, the idea of "distillations" describes the generation of surrogate models capable of answering concrete questions based on the collected data, e.g., to suggest mitigations of military attacks or to study the impact of cyber warfare [124]. Unsurprisingly, in recent works the generated surrogates have often taken the form of deep learning models. For instance, a simulation model of passengers moving in an airport was created based on empirical trajectories and subsequently exercised to generate passenger trajectories under various conditions to train a long short-term memory-based surrogate [256]. By this form of data augmentation, deep learning's requirement for vast amounts of data can be satisfied without surrendering the adherence to real-world data.

In many cases, gaining a more detailed understanding of regularities in the input-output relationship encoded in the model is desirable. Here, emerging XAI methods enable simulationists to leverage neural networks' ability to learn compressed representations of complex, non-linear relationships. By first training a neural network on simulation input-output pairs and applying XAI methods such as SHapley Additive exPlanations [184] to the trained network, the importance of different "features" representing the simulation model parameters is revealed [88]. In this setting, automated machine learning (autoML) has been applied to compare different types of surrogates without the need for manual tuning of the models' hyperparameters [260].

• **Uncertainty Quantification (UQ)**. A clear characterization of the degree of uncertainty in a model is key to establish an appropriate level of trust in the generated predictions. The sources of uncertainty in a model can be classified as aleatoric (caused by inherent stochasticity) and epistemic (reflecting gaps in the knowledge about the system, e.g., due to sparse or noisy data). Simulation-based inference, as discussed above, offers the advantage that the posterior describes the uncertainty about the parameters in a principled way. Dedicated UQ methods usually rely on statistics determined based on large numbers of model evaluations. Given the computational intensity of many simulation models, it is natural that various forms of neural surrogate models have been proposed [281, 146].

Ensemble approaches to UQ characterize epistemic uncertainty based on the variation among predictions generated by different models created from the same data [57]. Here, recent work proposed substituting entire ensembles with a neural network, achieving high-quality estimations of output standard deviations in benchmark problems [254].

A use case for learning the error between high-fidelity data and mechanistic simulation models (cf. Section 3.2) is to determine tighter uncertainty bounds than those attained by classical UQ approaches. This idea has been applied to UQ for turbulent flow simulations using perturbation methods [62]. By controlling the perturbations using a deep learning model that corrects for the simulation error, originally overly conservative uncertainty bounds are refined.

Stochastic spectral methods for UQ approximate the simulation response surface by a linear combination of orthogonal basis functions determined based on sampled simulation outputs. The use of tensor methods has been proposed to improve the scalability of these approaches to high-dimensional parameter spaces [75]. Tensor methods extend matrix decomposition and completion techniques to higher dimensions, allowing partial tensors that reflect the sampled simulation data to be completed to form high-quality approximate surrogates using a limited number of samples. For instance, in an electronic design automation context, the effect of parameter variations applied to $n_1 \times n_2$ circuits within each of n_3 dies on a wafer can be represented as a three-dimensional tensor [182]. In this setting, relying on tensor completion reduced the execution time by two orders of magnitude compared to a baseline approach from signal processing, while retaining similar levels of accuracy.

• **Rare Event Simulation**. The goal of rare event simulation is to sample in a way that is likely to trigger certain occurrences of originally low probability, which traditionally is often achieved via *importance sampling* [167]. In recent works, neural importance sampling methods have been proposed in which the proposal distribution is generated by a neural network trained on the observed samples [204, 234, 102, 131]. This idea has been extended to provide statistical bounds on rare-event probabilities for safety-critical applications [20].

Table 1: Overview of emerging AI-assisted approaches in simulation. Circled numbers refer to the annotations in Figure 2. Legend: *DL: deep learning, RL: reinforcement learning, LSTM: long short-term memory, GNN: graph neural network, RNN: recurrent neural network, CNN: convolutional neural network, GAN: generative adversarial network, MCMC: monte carlo Markov chain, real NVPs: real non-volume preserving transformations, AD: automatic differentiation, IL: incremental learning, NARX: nonlinear autoregressive exogenous model, DS: differentiable simulation.*

Stage	**Task/Method**	**Role of AI**	**Techniques/Architectures**
Model creation	Process mining ①	Identify possible structures and mechanisms	*LLM*
	Model creation from natural language ①④⑤	Code generation	*Transformer*
Model Calibration	Simulation-based inference ③	Choose summary statistics for ABC	*DL*
		Learn approximate posterior, likelihood, or both	*DL, transformer*
		MCMC: Design transition kernel	*RL*
		Proposal distribution for MCMC	*DL*
		Form subposteriors to parallelize MCMC	*Real NVPs*
	Data assimilation ②	Estimate likelihood of change points	*DL, NARX, RNN*
		Reparametrize model	*Particle filter, SMC, MCMC, RNN*
		Update model logic and structure	*Decision trees*
		Provide neural error correction terms	*DL*
Joint model creation and calibration	Symbolic regression ②	Steer selection of sub-expressions	*RL, distribution learning*
		Directly generate expressions	*Transformer, Monte Carlo tree search*
	Program synthesis ②	Generate programs respecting mechanistic constraints	*Neurosymbolic programming, PIML*
Simulation experiments	Runtime support ⑥⑦	Algorithm selection, autotuning	*DL, RL*
	Surrogate modeling ⑥	Learn fast and differentiable neural surrogates	*DL, LSTM, GNN, CNN, transfer learning, active learning*
	Simulation-based optimization ⑦⑨	Accelerate model evaluations	*DL, ensemble modeling, model preemption, clustering, IL*
		Accelerate sampling for robust design automation	*Neural surrogates, GAN*
		Enable gradient-based search	*Neural surrogates, AD, stochastic gradient estimation, DS*
	Reinforcement learning ②⑦	Learn world models	*DL, RNN*
	Output analysis ⑧	Learn "distillations" from simulation outputs	*Data augmentation*
		Extract parameter importance via explainable AI	*SHapley Additive exPlanations*
		Learn output variation for uncertainty quantification	*DL*
		Steer sampling towards rare events	*DL, DS, autoencoder, GAN*

In molecular simulations, where a rare event of interest may be a transition between two stable states, neural networks have been applied to learn difficult-to-capture low-frequency movements of molecules [37]. In an iterative approach, the simulation is sampled and the training repeated to gradually improve both the neural approximation and the probability of sampling the rare event. Differentiable simulations (cf. Section 3.4.3) have been used to cast efficient rare-event sampling for chemical reaction simulations as a optimization problem solvable via gradient descent [265]. The combination of high-performance compute clusters and advanced sampling mechanisms has been used to uncover the infection mechanisms of the SARS-CoV-2 spike protein [50]. In this work, an autoencoder operating on 3D PointNets [238] classifies the observed system states to identify states that should be sampled next. Similarly, GANs were used to influence the potential energy surface among molecules to steer simulations towards a target distribution [319]. In this setup, a sampling engine draws from a current candidate distribution, while a discriminator attempts to determine whether the samples originate from the desired distribution.

3.5 Discussion

The above overview of existing uses of AI techniques across the M&S life cycle is summarized in Table 1. Revisiting the research architecture of Figure 2, the circled numerical annotations link the concrete AI-driven tasks and methods to the processes in the M&S life cycle that lead from one artifact to another. Considering the figure, an evident gap in the existing work is centered around domain theory: We are not aware of contemporary existing works that employ AI methods to generate simulation meta models from existing theory, or to extract propositions from theory to support the creation of conceptual models and to validate simulation outputs. While there is existing work aimed at virtually all other artifacts and processes shown in the research architecture, there are still critical limitations in terms of reliability, interpretability, integration among AI and simulation components, and achievable scales. We will explore these aspects in detail in Section 5 in order to formulate concise challenges, together with avenues for research that could close the observed gaps.

4 Heterogeneous and Emerging Computing Platforms for Modeling and Simulation

The design and implementation of efficient algorithms is tightly linked to the capabilities and constraints of the machines on which they are executed. Like in other fields, the advancements in computational methods to support M&S studies have thus closely tracked the hardware developments from the early mainframes of the 1940s to today's landscape of complex and heterogeneous compute devices. The past twenty years presented two major turning points that made it necessary for M&S algorithms to adapt to new hardware realities. Firstly, since around 2005, the difficulty of achieving ever-smaller MOSFET scales has forced a move towards multi-core and many-core processing [272]. Secondly, the continuing series of breakthroughs in AI of the past years has led to a strong emphasis of cloud and high-performance computing installations on accelerators for neural network training and inference. Hence, in terms of raw compute power, the hardware platforms available to M&S researchers are now dominated by graphics processing units (GPUs) rather than CPUs, with various non-traditional AI-focused accelerators on the horizon. In the following, we briefly chart the development of simulation algorithms throughout the past decades' evolution in hardware design before sketching existing uses and opportunities of the recent and ongoing developments towards AI-oriented compute platforms.

4.1 Balancing Hardware Abstraction and Exploitation

When designing modeling languages and simulation engines, there is a fundamental tension between the goal of abstracting from application and hardware properties on one hand, and full exploitation of the properties to maximize efficiency on the other. This tension is apparent starting from the very beginnings of general-purpose computing, when Von Neumann's improvements to the ENIAC computer made it possible to program the machine using switches instead of manually rewiring its functional units. However, while this change vastly reduced the programming efforts required for ENIAC's uses in ballistic simulations, it also disabled the machine's capability for executing instructions in parallel [294].

Simulation engines reside on this spectrum between generality and specificity, in many cases aiming at supporting a certain class of models and hardware at reasonable levels of efficiency. Given the dominance of single-core CPUs from the 1970s to the early 2000s, a substantial portion of the classical work in broadly applicable simulation algorithms focused on data structures for sequential CPU-based simulations [33, 266, 41, 262, 275, 190]. From the late seventies onwards, a wealth of research in parallel and distributed simulation enabled researchers to harness the growing availability of multi-processor machines, multi-core processors, and local-area or wide-area computer networks. This line of research gave rise to algorithms such as Chandy & Misra's NULL messages [52] or Jefferson's Time Warp [139]. Later successors to these algorithms such as YAWNS [211] or Breathing Time Buckets [268] cater to the fact that the efficiency of different synchronization algorithms is model-dependent [94]. Considering this groundwork, the M&S world was well-positioned to tackle the 2000s rapid shift from single-core to multi-core hardware. In 2013, Barnes et al [30] demonstrated superlinear speedup for the Time Warp approach using reverse computation [47] when executed on nearly 2 million cores of the IBM Blue Gene/Q supercomputer. Recent work on CPU-based parallel and distributed simulation focused on achieving high performance for particularly challenging types of workloads [135, 15, 231] and on dynamic algorithm selection to achieve efficiency for ranges of models [122, 14].

4.2 Graphics Processing Units

Early harbingers of today's AI-driven push towards heavily GPU-focused compute infrastructure can be observed since about 2006, when GPUs first started to be widely used for various general-purpose tasks, and increasingly for scientific simulations. Modern GPUs are massively parallel processors following a largely single-instruction multiple-data (SIMD) architecture, owing to their heritage as accelerators for visualization tasks. Their sheer computational power is demonstrated by the fact that many of the supercomputers of the Top500 list obtain the majority of their reported (Exa-)FLOPS from commodity or custom-designed GPUs. Although recent architectures feature improved capabilities for handling divergence control flow within each SIMD unit, GPUs still favor computational tasks that can be decomposed into thousands of largely independent operations. Hence, they lend themselves to uses as co-processors to accelerate specific portions of simulation models that involve expensive, yet readily parallelizable, calculations such as those found in detailed simulations of wireless communications [25, 10].

By running large ensembles of simulation runs in parallel [141, 159, 163], GPUs can vastly accelerate data farming experiments, optimization methods such as genetic algorithms, large sampling experiments for rare event simulation, or perturbation analyses for uncertainty quantification. While the parallelization across independent simulation runs can benefit from GPU-specific optimizations [163], the fundamental algorithms can still largely follow their CPU-based counterparts.

More profound algorithmic adjustments are required when aiming to accelerate an individual simulation run using GPUs. While model-specific implementations of individual simulation models can achieve significant speedup [226, 90, 95, 298], a more generic approach is to design GPU-specific variations and specializations of CPU-based algorithms for parallel and

distributed simulation [225, 218, 179, 248, 9], which have proven to achieve order-of-magnitude accelerations and more over their CPU counterparts.

Another line of research applies relaxations to express simulation models as computational graphs [7, 61], originally motivated by the goal of facilitating automatic differentiation. If explicit parameter-dependent control flow in the form of discrete branches is eliminated through model-specific relaxations, a state update from one point in simulation time to the next often takes the shape of a largely smooth function over $\mathbb{R}^N$, where N is the number of state variables. Aside from facilitating gradient estimation, an equally important benefit of the resulting model formulations is their suitability for execution on GPUs, which favor single-instruction, multiple-data workloads with minimal branching. In addition, the relaxed model can be executed directly via machine learning frameworks such as Tensorflow or Torch, enabling their direct integration into AI training and inference pipelines.

4.3 Neural Processing Units

Further opportunities for accelerated computing are offered by Neural Processing Units (NPUs), which in recent years have found their way into high-performance computing installations and data centers [56]. These specialized chips follow architectures optimized to the needs of neural network training and inference, i.e., matrix multiplication, element-wise multiplication and addition over vectors, and vectorized application of activation functions. Among the typical features of NPUs is the support for efficient operations on reduced-precision floating point number representations using 16 and fewer bits. For instance, Google's Tensor Processing Units have been shown to outperform high-end GPUs both in terms of computational speed and in power consumption in the MLPerf training benchmark [147].

4.4 Field-programmable Gate Arrays

Reconfigurable hardware in the form of Field-Programmable Gate Arrays (FPGAs) adds the opportunity to synthesize specialized logic for a given problem. Since FPGAs often outcompete GPUs in terms of latency and power consumption, they are particularly relevant for handling real-time data. NPU architectures like Microsoft Brainwave [63] can be synthesized onto FPGAs, allowing the datapaths and precision to be tailored to a given neural network architecture [39]. The ability of FPGAs to accelerate simulations has been shown in the context of traffic simulations [300] and, more broadly, for discrete-event simulations [243]. Given their capability to achieve extremely low latencies and low energy consumption, FPGA-based simulations hold promise for prediction tasks and what-if analyses in the context of Digital Twins.

4.5 Cloud Computing

To make best use of cloud computing platforms, "simulation-as-a-service" frameworks have been proposed, which offer interfaces that largely abstract from the execution of the model on a suitably equipped machines as well as the collection and visualization of results [314, 276, 261]. In this context, the recent literature underlines the need for achieving reproducibility by making use of virtualization and containerization [201]. On the other hand, it has been observed that cloud computing's multi-tenancy configurations, in which several virtual machine's may share and dynamically migrate across physical machines, introduce challenges for synchronizing parallelized simulations, motivating virtualization-aware scheduling methods [313, 307]. To cater the non-uniform memory access of typical cloud servers, synchronization algorithms may also need to be adapted [223].

4.6 Neuromorphic Computing

Considering emerging and future changes in AI-oriented compute infrastructures, processing elements that depart from strictly Von Neumann-type architectures may soon find their place in the hardware mix readily available to researchers. For instance, neuromorphic chips, which mimic the structure of neurons in the human nervous system, are key enablers for large-scale experimentation in neuroscience on one hand [203], and AI tasks on the other [80, 200]. For example, the IBM NorthPole processor [200] obtains 25 times frames per second (FPS) per watt, and a 22 times lower latency than state-of-the-art GPU performance for the ResNet50 benchmark network. Closely integrating memory and compute elements, these chips are used to implement both traditional artificial neural networks and spiking neural networks, often in an approximate fashion. Beyond digital neuromorphic chips [97], analog designs [220] point towards the potential for a more widespread renaissance of analog computing, with recent successes demonstrating high efficiency for linear algebra operations in an AI context [5]. For a recent comprehensive review and discussion of neuromorphic computing "at scale", see [259, 162].

4.7 Bio-Computing

Another ongoing line of research is considering entirely biological forms of computing, using materials such as DNA and proteins as computational substrates [114, 105]. For instance, with DNA taking the role of storage and molecular circuits to implement logic gates, non-trivial programs can be realized [224]. A notable emphasis of recent research in bio-computing lies on implementing neural networks [302, 306]. Although the energy consumption of neural networks based on bio-computing compared to digital implementations is minimal, substantial hurdles such as the difficulties associated with introducing information into molecular circuits, retrieving computational results, handling noise, as well as minimizing bias must still be overcome before practicality for real-world applications can be achieved [306].

4.8 Quantum Computing

Finally, quantum computers are already being used experimentally and are expected to be able to simulate quantum phenomena such as quantum chemistry and quantum many-body problems as well as solve certain classes of optimization problems at vastly higher speed than traditional Von Neumann machines [250]. Given the probabilistic nature of quantum computers, the logic implemented in quantum circuits establishes a random distribution over possible outcomes, which is sampled from upon measurement. Under varying assumptions regarding the capabilities of practical quantum computers, quantum algorithms for linear algebra and various AI tasks have been proposed that would offer up to exponential speedup [84]. For optimization problems, quantum amplitude estimation (QAE) [40] has been shown to offer a quadratic speedup over algorithms on traditional architectures. Since QAE is equivalent in result to Monte Carlo estimation of a stochastic function's expectation, this benefit translates to simulation-based optimization problems as well [98, 76]. Hence, the availability of practical quantum computers would revolutionize M&S studies requiring high-dimensional parameter estimation and model calibration.

4.9 Exploiting Heterogeneous Hardware Platforms

Modern high-performance compute clusters and data centers already offer various combinations of CPUs, GPUs, FPGAs, and NPUs. As sketched above, each of these types of heterogeneous processing elements follows its own programming model and performs best for particular computations under various constraints in terms of memory requirements, types of instructions and precision, latency, and degree of parallelism. The AI field has responded to the need for abstracting from the underlying hardware by the emergence and sweeping success of machine

Stage	Challenge	Research Directions
Model Creation	Forming hypotheses on the mechanisms underlying complex processes	• *Relational learning and causal modeling* • *Rigorous logical reasoning*
Model Calibration	Exposing information on simulations' output-to-parameter relationships	• *Generalized and sample-efficient gradient estimation* • *Alternative modes of model exploration*
Joint Model Creation and Calibration	Model synthesis at scale	• *AI-driven search strategies* • *Accounting for explicit and implicit domain knowledge*
Simulation Experiments	Efficient execution of heterogeneous simulation/AI experiments on heterogeneous hardware	• *Interoperability among simulation and AI platforms* • *Experiment-aware job scheduling* • *Malleable Simulation Models*

Table 3: Research challenges on the path towards an intelligent modeling and simulation life cycle.

learning frameworks such as Tensorflow or PyTorch, which allow models and training procedures formulated in high-level languages to efficiently be mapped to the available hardware. In cloud environments, virtualization and containerization is widely used to lift dependencies of AI software pipelines on machine-specific software environments and to achieve reproducibility [257].

While the M&S field has not caught up with the ease of use of AI frameworks, several frameworks allow modelers to express simulation models in a hardware-agnostic way, largely shielding them from the complexities of the target platform [157, 248, 158, 212]. These frameworks generate hardware-specific code from high-level model descriptions in domain-specific modeling languages. This way, the same model description can be compiled for different hardware targets, e.g., for CPUs or GPUs. However, making full use of heterogeneous hardware platforms requires the combined use of different types of processing elements for different portions of a simulation [210]. An extension to the agent-based modeling language OpenABL [69] employs performance predictions and autotuning for this purpose, dynamically distributing different model portions to CPUs, GPUs and FPGAs, while accounting for communication costs [299].

Tomorrow's spectrum of compute hardware available for M&S studies will likely comprise a variety of digital and analog computing devices, which will be able to serve the needs of models at various levels of fidelity and speed. At the same time, this will call for model representations of various types, from traditional CPU-based models and data-parallel variants, to relaxed formulations with limited explicit control flow or neural surrogates trained purely on pairs of simulation inputs and outputs. The hurdles and research opportunities on the way towards making best use of heterogeneous models and platforms will be explored in Section 5.4.

5 Research Challenges and Roadmap for Intelligent Modeling and Simulation Life Cycle

The previous sections summarized the landscape of existing uses of AI approaches, techniques, and the emerging hardware landscape to support simulation studies. In the following, we revisit the individual steps of the M&S life cycle to identify research gaps in the form of four challenges which, if tackled, would lead to a deeper integration between the simulation and AI fields and to a higher degree of automation in simulation-oriented scientific experimentation and discovery. For ease of navigating the text, Table 3 summarizes the challenges and research directions detailed below.

5.1 Model Creation

As discussed in Section 3, the main techniques that support modelers in determining a model's *structure and logic* from data or informal descriptions fall under the categories of process mining or model creation from natural language. Of these, the former currently is largely focused on identifying processes about which substantial a priori knowledge is available, e.g., for determining the specific sequence of activities out of a well-understood pool of possible activities in a supply chain process. At the other extreme, model creation directly from natural language currently lacks the interpretability, reliability, and justification that would create sufficient trust in the generated models to apply them to real-world use cases. This is in line with the aforementioned lack of work on employing AI-driven automatic reasoning to deduce propositions from domain theory, and on evaluating data in light of these propositions (Section 3.5). We postulate that what is missing are reliable techniques to translate hypotheses gathered from data and theory into mechanisms executable as part of simulation models, corresponding to the processes "identification of structure and parameters" and "deductive reasoning" in Figure 2).

Challenge: Formulating justified hypotheses on the mechanisms underlying complex processes.

5.1.1 Research Direction: Relational Learning and Causal Modeling

Relational learning identifies interactions among entities in a graph and characterizes them quantitatively [155]. Such a representation can provide pointers towards mechanisms underlying the target system's behavior. The focus on interactions makes the approach particularly attractive for agent-based modeling [244], where the overall system behavior emerges from localized interactions among large numbers of agents [113]. Recent relational learning methods tend to employ neural networks, e.g., variational autoencoders (VAEs), in order to infer interactions from data [156] and have been applied to inferring interactions within biochemical systems [177, 311] and multi-agent systems in robotics [172].

In comparison to relational learning, which considers the general problem of identifying (any form of) dependencies between entities or variables, *causal modeling* aims to identify cause-and-effect relationships between variables and estimating the impact of interventions. Traditionally rooted in statistical analysis, it increasingly leverages machine learning techniques to infer causal relationships from observational data automatically. For example, VAEs and transformers can learn hidden patterns that capture the underlying causes in data. While transformers can leverage the attention mechanism to identify potential causal structures [195], VAEs focus on learning disentangled latent representations [305]. If each dimension in the latent space corresponds to an independent causal factor, then manipulating one dimension leads to predictable changes in the reconstructed data while other dimensions remain unaffected. This allows for the identification of potential causal relationships between the latent factors and the observed variables, known as causal discovery. Reinforcement learning has also been used to explicitly identify the underlying causal structure of a system, where an agent learns to navigate the space of causal graphs, whose nodes and directed edges represent cause-and-effect relationships [329]. Once the causal structure is inferred, variational inference can be used for causal inference [100]. By intervening on specific dimensions of the latent space (i.e., changing their values), we can observe how these interventions propagate through the decoder and affect the reconstructed data. This enables the estimation of the causal effects of interventions on the observed variables.

The insights gained from relational learning and causal modeling can be viewed as justified hypotheses on interactions and cause-and-effects relationships, which modelers can draw on as building blocks when constructing mechanistic models.

5.1.2 Research Direction: Rigorous Logical Reasoning

Even with the early successes of using LLMs to support modelers in identifying and formalizing hypotheses on mechanisms as a basis for simulation models, the approach still suffers from current LLMs' limitations in handling tasks that require rigorous forms of reasoning [286, 21, 267]. To allow modelers to rely on the identified model building blocks, they must be able to trust that the mechanisms are not in violation with established facts about the system to be modeled.

Prompt engineering is a common technique to enhance the reasoning abilities of LLMs without requiring additional training steps or modifications to the LLM's architecture. A prominent example is chain-of-thought (COT) prompting [293], in which the LLM is prompted to state the full sequence of logical steps taken in a reasoning task. COT prompting has been shown to improve the coherence of LLMs in various benchmarks focused on reasoning. Various subsequent works have proposed prompting techniques that guide an LLM to lay out its reasoning in a step-by-step manner using semi-formal or formal descriptions [43, 175, 303].

The adherence of LLMs to an existing knowledge base can also be improved through targeted fine-tuning [45]. By gradually maximizing the probability the LLM assigns to a set of facts, the LLM is grounded in an existing knowledge base.

A line of research that departs from purely neural architectures is neurosymbolic AI [99]. The fruitful combination of neural and symbolic model elements has been studied for decades, but the enormous successes and observed limitations of LLMs have created an explosion of research towards this goal, which has been termed the "the third wave of AI" [99]. Here, we are focused on LLM-oriented work that may permit a direct interaction with modelers in the future. Relevant work outside the LLM realm is discussed in Section 5.3.2.

Many recent works focus on coupling LLMs with solvers for various logics in order to introduce rigorous reasoning capabilities. Typically, this takes the form of a feedback loop [214, 213]: first, a user-provided prompt in natural language is translated by an LLM to a logic problem, e.g., in terms of first-order logic, constraint satisfaction, or satisfiability. The solver's output is fed back to the LLM, which generates a user-readable conclusion or continues the cycle to conduct further reasoning. Error messages from the solver may also be fed back to the LLM to allow for iterative refinement or correction of the problem formulation.

Considering the time-consuming nature of model creation as part of the M&S life cycle, automatically verifying the plausibility of mechanistic hypotheses could vastly accelerate this initial phase of simulation studies. For static properties, this could take the form of integrating LLMs with logic solvers as described above. By the use of simulators driven by suitable formal modeling languages, it would further be possible to evaluate dynamic properties and thus also to automatically generate mechanisms that produce certain expected outcomes. We are not aware of any works that explore this form of neurosymbolic AI for simulation model creation.

5.2 Model Calibration

In present scientific studies, neural models and simulation models are often still treated as fundamentally different categories. Among the reasons for this delineation is the neural models' data-driven creation. While the choice of architectures and hyper-parameters of neural networks do allow prior knowledge and constraints to be encoded, the key enabler for these models' sweeping successes is the directed search for a "locally optimal" fit via gradients computed using backpropagation. Derivatives, which can provide important indications for the relationships between the model outputs and the inputs, are not easily available for many types of simulations, with hurdles ranging from the purely technical [1] to more fundamental difficulties such as the joint occurrence of stochasticity and discrete control flow [22, 160].

In line with other authors [165], we believe that departing from the traditional view of simulation-as-blackbox will help close the gap between the AI and simulation world by enabling more tightly integrated neural/simulation models and more efficient experimentation, e.g., by a joint calibration and training of simulation models involving neural components.

Challenge: Exposing information on simulations' output-to-parameter relationships.

5.2.1 Research Direction: Generalized and Sample-Efficient Gradient Estimation

The local sensitivity information provided by partial derivatives is useful not just for calibration, but also for general optimization, sensitivity analysis, and uncertainty quantification. Often, the calculated gradients are those of the expectation of a stochastic simulation output with respect to the parameters. As discussed in Section 3.4.3, a number of specialized or generic estimators exist that apply automatic differentiation (AD) to simulation models, often based on relaxations or restrictive assumptions to handle discrete model elements. An alternative are stochastic black-box estimators such as those from the REINFORCE family [321], which are unbiased if parametrized for the model's underlying distribution, or generic estimators that apply smoothing to the original function [208]. Unfortunately, these black-box estimators often suffer from high variance, requiring tailored variance reduction methods, e.g., control variates, to be competitive with AD-based approaches.

Calibration and inference tasks could be vastly accelerated by efficient, i.e., low-variance and generic methods to calculate gradients of expectations, but also of other statistics such as higher moments, likelihoods, or extrema of state variables. One of the main challenges lies in capturing the regularities in the often highly variable behavior of stochastic simulations, even at a fixed parameter combination, without the need for extensive sampling. AI methods can play an important role in interpolating between the sampled points in the output space. For instance, local neural surrogates could be trained to predict the distribution of simulation outputs at a given point in the input space from a few samples gathered by simulation. Beyond neural networks, representing the gathered high-dimensional samples in a sparse tensor could allow tensor completion methods [75] to "fill in the blanks".

5.2.2 Research Direction: Alternative Modes of Model Exploration

Determining whether a model is consistent with available data may not only require a reasonable fit of an output distribution's shape to available data, but also guarantees that certain dynamic properties are upheld. Methods from (often stochastic) model checking [65] are applied to determine whether user-provided propositions are satisfied. This is achieved either by repeated executions of the unmodified model, or by different forms of abstract interpretation [70], which, instead of executing the program over concrete states, evaluates the program over abstract states, e.g., intervals that variable values may lie in. However, even in the deterministic case, providing guarantees that certain states are unreachable can easily become intractable beyond low-dimensional state spaces and short trajectories.

Abstract interpretation has also been used to approximate the expectation of a program's output distribution given an input distribution [54]. The resulting smoothed approximation of the input-output relationship facilitates calibration through numerical optimization. Again, the challenge is to capture the vast number of possible state trajectories of non-trivial programs at sufficient accuracy without exceeding memory and execution time limits.

A similar issue arises in simulations of quantum circuits [148], where each gate propagates a distribution of possible states whose dimensionality grows exponentially with the number of qubits. Here, probabilistic modeling via transformers has been employed to counter the state space explosion [48]. Applying such probabilistic approaches to model checking could enable analyses at larger scales, but will require careful consideration of the possible loss in fidelity or subsequent verification using traditional methods.

Finally, methods from reversible computing [227] have been used to avoid the large number of forward simulations required to establish whether certain model states are reachable. By reversing the model logic, the simulation can be initialized at a known or (un-)desired target output state and executed backwards in time to determine starting conditions that may lead

to the configured output states. The utility of this approach has been demonstrated in the context of road traffic and wafer fabrication simulations [12, 169]. Since far from all operations found in typical simulation models are bijections, the reverse execution involves branching points at which there are two or more valid previous simulation states. Efficient strategies to solve the resulting search problem are likely to be model-dependent, as knowledge of the model logic may rule out many reverse trajectories without requiring a full exploration. The recent successes in training problem-specific deep learning-based search heuristics for combinatorial optimization [149] make it plausible that learned heuristics could outperform generic or manually constructed strategies for reverse exploration of simulation models as well. In this field, there has recently been significant interest in deep learning-based methods, in particular for learning heuristic search policies [192]. These neural heuristics have been shown to excel both for classical tasks such as the traveling salesman problem and for more domain-specific ones such as vehicle routing [323].

5.3 Joint Model Creation and Calibration

As discussed in Section 3.3, there have been promising early successes in automatic generation of the structure and parameters of entire simulation models from data. However, at the time of writing, these approaches have been demonstrated for models of relatively modest scale [191, 161]. Achieving the lofty goal of real-world practicality, new methods will be required to tackle the vast search space of candidate models over combinations of discrete and continuous decision variables. It seems likely that intelligent and problem-specific search strategies will be needed, combined with a close integration of explicit, and possibly also implicit, domain knowledge to constrain the search space.

Challenge: Model synthesis at scale.

Considering the research architecture from Figure 2, this challenge concerns the automation of instantiating a domain theory and of implementing the resulting conceptual model.

5.3.1 Research Direction: AI-driven Model Search Strategies

The past years have seen promising advances in system identification for model classes such as population-based chemical reaction networks [161] and multibody systems [24]. The search across candidate model structures is often achieved by meta-heuristics such as genetic algorithms or particle swarm optimization [3].

A different problem formulation casts the identification of chemical reaction networks that can reproduce available data as a tree search [191]. In this approach, the nodes on each level of the search tree correspond to candidates for one additional reaction, which is chosen as the reaction that minimizes the variance between selected points on the empirical and simulated state trajectories. A subsequent classical mathematical optimization calibrates the tentative model generated so far to the observations.

The formulation as a tree search problem opens up the opportunity to employ different search strategies. As mentioned above, deep-learning based heuristics have been shown to be competitive in solving several combinatorial optimization problem [192]. Model synthesis is a natural application area for these approaches, where capturing some of the regularities that govern the link between model elements and outputs could provide important guidance towards well-fitting models.

5.3.2 Research Direction: Accounting for Explicit and Implicit Domain Knowledge

Beyond efforts towards a more directed model search, reducing the search space would also accelerate model synthesis and extend its reach towards larger models. A second reason for

reducing the search space is to counter the issue of non-identifiability [285]. In many cases, there are various models that are equivalent in their ability to generate outputs in line with the available data. When considering simulation models, this is particularly problematic because of their dual role not just as prediction engines, as in identification via neural networks [232], but also as interpretable manifestations of plausible mechanisms for the data-generating process.

To improve both the tractability and identifiability, it is key to rule out candidate models that are implausible or violate problem-specific properties outside the mere fit to the observations. This perspective changes the approach from black-box to grey-box system identification [180]. A challenge lies in obtaining and formalizing model constraints, which may stem from a general domain theory, assumptions by the modelers, or even commonsense reasoning. While in conflict with a full automation of the model synthesis process, a "skeleton" of a conceptual model that explicates known properties of the model could form a basis for restricting the search space. Here, LLMs can play two important rules. Firstly, translating from descriptions of model constraints specified in natural language to formal constraints applicable during automatic model search would allow non-technical domain experts to specify their assumptions in an informal manner. Secondly, by deriving implications from the stated domain knowledge, determining probable unstated assumptions, as well as augmenting this information using LLMs' capabilities for commonsense reasoning [324], the search space can be further reduced. Constraining the model search in this manner bears similarities to physics-informed learning approaches (cf. Section 3.3), but relies on constraints automatically determined from explicit or implicit sources and targets the creation of purely mechanistic simulation models.

Importantly, the generated constraints can be justified to the modeler in natural language. Integrating LLMs with rigorous logical reasoning capabilities, as described in Section 5.1.2, would reduce the need for verification and increase trust in the correctness of the search space reductions.

5.4 Simulation Experiments

Given the trends outlined in Section 3, there is a clear trajectory towards complex experiments involving combinations of AI and simulation tasks, one evident example being reinforcement learning over simulated environments as commonly used in robotics. However, the increasing use of AI to form surrogate models, to steer sampling and optimization processes, and for output analysis creates tasks comprised of individual steps that are heterogeneous in their computational characteristics and resource requirements. Considering the equally heterogeneous nature of cloud and high-performance computing environments available to researchers today, achieving high utilization puts new demands on the interface between the software and hardware side of simulation studies.

Challenge: Efficient execution of heterogeneous simulation/AI experiments on heterogeneous hardware.

5.4.1 Research Direction: Interoperability Among Simulation and AI Platforms

Over the years, both academia and industry have proposed various simulation runtime environments, each characterized by distinct capabilities, performance profiles, and hardware compatibilities. The performance requirements of simulation tasks vary over the simulation life cycle, e.g., model calibration may require numerous simulation runs at small scenario scales, whereas applying the simulation model for a real-world purpose may require only a few simulations runs at large scenario scales. To cater to this variability in requirements, reducing the overall time to complete the required model executions over the stages of an entire simulation life cycle would require switching among simulation frameworks or executing several in conjunction. Although existing architectures and standards support the coupling of simulators in a principled manner [77, 32], there is a lack of generic mechanisms for targeting multiple simulators using the same code, and for transferring dynamic model states across simulators. Such capabilities would

not only streamline the modeling process but also enhance the reproducibility and portability of simulation experiments. By focusing on cross-platform compatibility, researchers and practitioners can leverage the strengths of different runtime environments, optimizing performance and resource utilization without compromising on the ease of model development and deployment. Making this possible will require new techniques and tools that enable model developers to formulate their models and execute them across multiple platforms without modification. Code generation from domain-specific languages is customary in several M&S paradigms [233] and application domains [308, 119, 152], with recent progress towards supporting heterogeneous architectures and more broadly defined classes of agent-based models [299, 248].

To broaden the scope beyond simulation models, towards experiments involving simulation and AI aspects, it is critical to take into account the interactions between the two realms and the specific performance profiles and compatibilities of AI software components. If simulation/AI experiments are modeled as a unit, the code generation can maximize not only the efficiency of the simulation and AI components when executed together, but also the interfacing between the two. This enables a loose coupling among components to be retained on a conceptual level for maintainability, while the generated code may employ shared memory or device-specific capabilities, such as remote direct memory access, to minimize the overhead for inter-component communications [217]. As a simple example, during training of a neural surrogate model, placing both the simulation model state and the neural network's parameters in GPU memory could reduce the need for expensive GPU-to-host data transfers required in a traditional loose coupling between a simulator and an AI framework such as Torch.

To achieve this ambitious goal, we propose the adoption of model-driven engineering (MDE) [251] techniques beyond the well-established reliance on modeling languages in M&S. MDE facilitates the creation of platform-independent models that can be systematically transformed into various platform-specific implementations, ensuring consistent and accurate communication between disparate systems [189]. This approach mitigates the challenges of manually re-implementing simulation models for different platforms, reducing errors and enhancing maintainability.

By employing standardized modeling languages and tools, MDE enables the seamless integration of heterogeneous simulation environments. Building on existing efforts in modeling the activities and artifacts involved in simulation experiments [295], metamodels and domain-specific languages will be needed to explicitly specify the involved AI and simulation components, their performance profiles, as well as their interactions.

5.4.2 Research Direction: Experiment-Aware Job Scheduling

Given the opportunity to assign simulation and AI tasks to various combination of hardware devices, an optimal assignment is heavily dependent on the overall experiment. As an example, consider the use of active learning to train a neural surrogate for a simulation model (cf. Section 3.4). In this setting, neural network training steps are interleaved with simulation runs to gather additional information at dynamically identified points in the parameter space. This setup combines the dense and predictable computational structure of the forward and backward passes during neural network training with the sparse and highly data-dependent computations of typical simulations. The former is well-served by the execution on a GPU, whereas the latter may benefit more from the highly advanced capabilities of modern CPUs to handle complex control flow. However, the need for data exchange between the simulation and the neural network training may still lead to the a purely GPU-based execution being the most efficient option.

More broadly, as different types of simulation experiments come with different combinations of tasks and varying opportunities for parallel or distributed execution (e.g., generations of simulation runs when optimizing using genetic algorithms), there is a clear need for experiment-aware job scheduling mechanisms capable of making the best use of the available hardware. By exploiting knowledge of the computational patterns associated with different experiments,

job scheduling and hardware assignment mechanisms can better predict the static or dynamic resource demands and thus make decisions closer to the optimum. Recent work in deep learning-specific job scheduling [309] can be a stepping stone towards this vision, but will need to be augmented to tackle the characteristically high dynamism in simulation tasks and their resource and time demands, as well as the vastly increased space of candidate task-to-device mappings. In the past years, various approaches for machine learning-supported job scheduling have been proposed, typically combining reinforcement learning-based scheduling policies with dynamic predictions of job execution times based on input parameters and intermediate outputs [86, 316, 290, 206]. Emerging approaches towards automating the design of simulation experiments [296] can help in gathering the information required to find hardware assignments and schedules that minimize a study's "time-to-insight".

5.4.3 Research Direction: Malleable Simulation Models

In the presence of a variety of heterogeneous compute devices, it is desirable to allow users to run their simulation models on any of the available and unoccupied devices, as long as the achievable fidelity suffices for the purposes of the present study. We have seen in Section 4 that specialized and analog accelerators are entering the hardware landscape offered to researchers. Hence, achieving a transparent hardware mapping requires high-level model formulations that allow not just for a hardware mapping across digital computing devices, but that can compile models to traditional machine code as well as forms suited for execution on analog hardware. For instance, when targeting neuromorphic hardware, this could mean translating models to neural networks. A direct translation of this sort has been achieved for logic formulae in the context of neurosymbolic AI [249]. In the simulation realm, works proposing smoothed model approximations are stepping stones towards this goal, but are currently confined to specific classes of models [8, 61]. Error bounds or empirical results will be needed to judge whether the error created by such relaxations can be tolerated.

The capability for targeting hardware platforms at different levels of fidelity and speed opens up new opportunities for optimizing the integration of simulation runs into larger experiment setups. In experiments where sampled simulation outputs serve as input to learning, as is often the case in reinforcement learning and surrogate modeling, simulations at the full fidelity of an original simulation model may not always be needed. For instance, at the early stages of a reinforcement learning procedure, actions may be largely random and thus comparatively rough reward estimates may suffice to make progress, whereas towards the end of a training procedure, small variations in reward may be decisive to fine-tune the reinforcement learning agent's behavior.

Regardless of whether the available model variants stem from classical surrogate modeling or are generated from a high-level model description as sketched above, a negotiation between the "learning" and the simulation parts of an experiment's feedback loop will be needed to maximize the key metric of learning progress per unit (wall-clock or compute) time. On the AI side, problem-specific estimation or, more generally, AutoML techniques [120] could determine the required level of fidelity at the current learning stage. On the simulation side, the problem presents as a form of model selection [81], the key requirement being an accurate characterization of each representation's fidelity, either by static analysis or dynamically through measurement.

6 Conclusions

Our aim is for this article to serve as a snapshot of the existing ways AI is supporting and enhancing the various tasks within the simulation life cycle, and to anticipate future opportunities and challenges towards this goal. Taking a step back from specific problem settings and techniques, most of the existing work broadly relies on two capabilities of AI techniques: function and distribution approximation via deep neural networks (e.g., of simulation input-output

relationships, likelihood functions, error terms), and the translation from unstructured or informal data to formal representations (e.g., determining causal relationships from data, generating numerical or mechanistic model building blocks from natural language). While neural approximations and surrogates have been studied for decades and have been fruitfully applied at nearly all stages of the simulation life cycle, the generative abilities of recent AI architectures point towards an increasing amount of automation of previously purely human-driven steps in scientific studies.

We believe that the main hurdles towards a closer integration between AI and simulation pertain to two main aspects. First, many conceivable use cases for AI in simulation studies are unlikely to take root without offering the guarantees, well-justified sequences of steps, or proofs obtained by rigorous logical reasoning. While important ongoing work explores the limits of such capabilities within current state-of-the-art models' probabilistic and inductive type of reasoning, a turn towards neurosymbolic AI seems inevitable to credibly produce and assess artifacts involved in simulation studies of real-world criticality. Second, and more broadly, a shift is needed towards unified AI/simulation models and experiments. Given the reality that mechanistic and neural models are now routinely coupled in AI training loops, used as problem-dependent substitutes in surrogate modeling, and combined to form hybrid models, the traditional delineation between AI pipelines and simulation workflows or models is becoming less tenable. In this light, it is essential to expose, and subsequently fulfill, the respective requirements of AI and simulation aspects of a study, be it through a closer integration of models via differentiable simulation models, better utilization of AI-oriented compute platforms by hardware-agnostic model formulations, or by experiment-aware hardware assignment and scheduling methods.

The authors hope that by sketching pathways towards such a coalescence of AI and simulation techniques, this article will contribute to accomplishing the vision of an intelligent M&S life cycle.

Acknowledgments

The authors express their gratitude to Schloss Dagstuhl for hosting the Dagstuhl Seminar 22401, *Computer Science Methods for Effective and Sustainable Simulation Studies*, where the initial ideas for this article were conceived.

Funding

PA and WC are supported by the National Research Foundation, Singapore under its AI Singapore Programme (AISG Award No: AISG3-RP-2022-031).

Initial work by PA was funded by Deutsche Forschungsgemeinschaft (DFG, German Research Foundation), grant no. 497901036.

AP has been partially supported by the Italian MUR PRIN 2022 Project: Domain (Grant #2022TSYYKJ, CUP E53D23008200006, PNRR M4.C2.1.1).

CDC was supported by the U.S. Department of Energy (DOE) under Contracts: No. DE-AC02-06CH11357, the "Kronos: Enabling Long Timescale PDES Simulations via Multi-Resolution Methods" project, and No. DE-SC0024616, the "Tachyon: Intelligent Multi-Scale Modeling of Distributed Resilient Infrastructure and Workflows for Data Intensive HEP Analyses" project.

VW also received support from the German Federal Ministry of Education and Research (BMBF) as part of the project MAC-MERLin (Grant Agreement No. 01IW24007).

References

[1] Max Aehle. *Automatic Differentiation of Compiled Programs*. PhD thesis, Rheinland-Pfälzische Technische Universität Kaiserslautern-Landau, 2024.

[2] Ali Akhavan and Mohammad S Jalali. Generative ai and simulation modeling: how should you (not) use large language models like chatgpt. *System Dynamics Review*, 2023.

[3] Alireza Alfi and Hamidreza Modares. System identification and control using adaptive particle swarm optimization. *Applied Mathematical Modelling*, 35(3):1210–1221, 2011.

[4] Mohammed AlQuraishi and Peter K Sorger. Differentiable biology: using deep learning for biophysics-based and data-driven modeling of molecular mechanisms. *Nature methods*, 18(10):1169–1180, 2021.

[5] S. Ambrogio, P. Narayanan, and A. Okazaki et al. An analog-ai chip for energy-efficient speech recognition and transcription. *Nature*, 620:768–775, 2023. doi: https://doi.org/10.1038/s41586-023-06337-5.

[6] Samaneh Aminikhanghahi and Diane J Cook. A survey of methods for time series change point detection. *Knowledge and information systems*, 51(2):339–367, 2017.

[7] Philipp Andelfinger. Differentiable agent-based simulation for gradient-guided simulation-based optimization. In *Proceedings of the 2021 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*, SIGSIM-PADS '21, page 27–38, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450382960. doi: 10.1145/3437959.3459261. URL `https://doi.org/10.1145/3437959.3459261`.

[8] Philipp Andelfinger. Towards differentiable agent-based simulation. *ACM Trans. Model. Comput. Simul.*, 32(4), jan 2023. ISSN 1049-3301. doi: 10.1145/3565810. URL `https://doi.org/10.1145/3565810`.

[9] Philipp Andelfinger and Adelinde M Uhrmacher. Synchronous speculative simulation of tightly coupled agents in continuous time on cpus and gpus. *Simulation*, 100(1):5–21, 2024.

[10] Philipp Andelfinger, Jens Mittag, and Hannes Hartenstein. Gpu-based architectures and their benefit for accurate and efficient wireless network simulations. In *2011 IEEE 19th Annual International Symposium on Modelling, Analysis, and Simulation of Computer and Telecommunication Systems*, pages 421–424. IEEE, 2011.

[11] Philipp Andelfinger, Sajeev Udayakumar, Wentong Cai, David Eckhoff, and Alois Knoll. Model preemption based on dynamic analysis of simulation data to accelerate traffic light timing optimisation. In *2018 Winter Simulation Conference (WSC)*, pages 652–663. IEEE, 2018.

[12] Philipp Andelfinger, Jordan Ivanchev, David Eckhoff, Wentong Cai, and Alois Knoll. From effects to causes: Reversible simulation and reverse exploration of microscopic traffic models. In *Proceedings of the 2019 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*, pages 173–184, 2019.

[13] Philipp Andelfinger, David Eckhoff, Wentong Cai, and Alois Knoll. Fast-forwarding of vehicle clusters in microscopic traffic simulations. In *Proceedings of the 2020 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*, pages 63–66, 2020.

[14] Philipp Andelfinger, Andrea Piccione, Alessandro Pellegrini, and Adelinde Uhrmacher. Comparing speculative synchronization algorithms for continuous-time agent-based simulations. In *Proceedings of the 26th International Symposium on Distributed Simulation and Real Time Applications*, DS-RT '22, pages 57–66, Piscataway, NJ, USA, September 2022. IEEE. doi: 10.1109/DS-RT55542.2022.9932067.

[15] Philipp Andelfinger, Till Köster, and Adelinde Uhrmacher. Zero lookahead? zero problem. the window racer algorithm. In *ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*, PADS '23, pages 1–11, New York, NY, USA, June 2023. ACM. doi: 10.1145/3573900.3591115.

[16] Dimitrios Angelis, Filippos Sofos, and Theodoros E Karakasidis. Artificial intelligence in physical sciences: Symbolic regression trends and perspectives. *Archives of Computational Methods in Engineering*, 30(6):3845–3865, 2023.

[17] Dylan M Anstine and Olexandr Isayev. Machine learning interatomic potentials and long-range physics. *The Journal of Physical Chemistry A*, 127(11):2417–2431, 2023.

[18] Thomas Anthony, Zheng Tian, and David Barber. Thinking fast and slow with deep learning and tree search. *Advances in neural information processing systems*, 30, 2017.

[19] Rossella Arcucci, Jiangcheng Zhu, Shuang Hu, and Yi-Ke Guo. Deep data assimilation: integrating deep learning with data assimilation. *Applied Sciences*, 11(3):1114, 2021.

[20] Mansur Arief, Yuanlu Bai, Wenhao Ding, Shengyi He, Zhiyuan Huang, Henry Lam, and Ding Zhao. Certifiable deep importance sampling for rare-event simulation of black-box systems, 2021. URL `https://arxiv.org/abs/2111.02204`.

[21] Konstantine Arkoudas. Gpt-4 can't reason. *arXiv preprint arXiv:2308.03762*, 2023.

[22] Gaurav Arya, Moritz Schauer, Frank Schäfer, and Christopher Rackauckas. Automatic differentiation of programs with discrete randomness. *Advances in Neural Information Processing Systems*, 35:10435–10447, 2022.

[23] Gaurav Arya, Moritz Schauer, and Ruben Seyer. Gradient estimation via differentiable metropolis-hastings, 2024. URL `https://arxiv.org/abs/2406.14451`.

[24] Ehsan Askari and Guillaume Crevecoeur. Evolutionary sparse data-driven discovery of multibody system dynamics. *Multibody System Dynamics*, 58(2):197–226, 2023.

[25] Scott Bai and David M Nicol. Gpu coprocessing for wireless network simulation. In *Symposium on Application Accelerators in High Performance Computing, 2009 (SAAHPC'09*, 2008.

[26] Pradeep Bajracharya, Javier Quetzalcóatl Toledo-Marín, Geoffrey Fox, Shantenu Jha, and Linwei Wang. Feasibility study on active learning of smart surrogates for scientific simulations, 2024. URL `https://arxiv.org/abs/2407.07674`.

[27] Osman Balci. A life cycle for modeling and simulation. *Simulation*, 88(7):870–883, 2012.

[28] Yejin Bang, Samuel Cahyawijaya, Nayeon Lee, Wenliang Dai, Dan Su, Bryan Wilie, Holy Lovenia, Ziwei Ji, Tiezheng Yu, Willy Chung, et al. A multitask, multilingual, multimodal evaluation of chatgpt on reasoning, hallucination, and interactivity. *arXiv preprint arXiv:2302.04023*, 2023.

[29] Jerry Banks, John Carson, Barry L. Nelson, and David Nicol. *Discrete-Event System Simulation (4th Edition)*. Prentice Hall, 4 edition, December 2004. ISBN 0131446797. URL `http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20\&path=ASIN/0131446797`.

[30] Peter D. Barnes, Christopher D. Carothers, David R. Jefferson, and Justin M. LaPre. Warp speed: executing time warp on 1,966,080 cores. In *Proceedings of the 1st ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*, SIGSIM PADS '13, page 327–336, New York, NY, USA, 2013. Association for Computing Machinery. ISBN 9781450319201. doi: 10.1145/2486092.2486134. URL `https://doi.org/10.1145/2486092.2486134`.

[31] Russell R. Barton and Martin Meckesheimer. Chapter 18 metamodel-based simulation optimization. In Shane G. Henderson and Barry L. Nelson, editors, *Simulation*, volume 13 of *Handbooks in Operations Research and Management Science*, pages 535–574. Elsevier, 2006. doi: https://doi.org/10.1016/S0927-0507(06)13018-2. URL https://www.sciencedirect.com/science/article/pii/S0927050706130182.

[32] Jens Bastian, Christoph Clauß, Susann Wolf, and Peter Schneider. Master for co-simulation using fmi. In *8th International Modelica Conference*, volume 2011. Linköping University Electronic Press, Linköpings universitet Dresden, Germany, 2011.

[33] Rudolf Bayer. Symmetric binary B-trees: Data structure and maintenance algorithms. *Acta informatica*, 1(4):290–306, 1972. ISSN 0001-5903,1432-0525. doi: 10.1007/bf00289509.

[34] Alessandro Berti, Daniel Schuster, and Wil MP van der Aalst. Abstractions, scenarios, and prompt definitions for process mining with llms: a case study. In *International Conference on Business Process Management*, pages 427–439. Springer, 2023.

[35] Alessandro Berti, Humam Kourani, Hannes Hafke, Chiao-Yun Li, and Daniel Schuster. Evaluating large language models in process mining: Capabilities, benchmarks, evaluation strategies, and future challenges. *arXiv preprint arXiv:2403.06749*, 2024.

[36] Luca Biggio, Tommaso Bendinelli, Alexander Neitz, Aurelien Lucchi, and Giambattista Parascandolo. Neural symbolic regression that scales. In *International Conference on Machine Learning*, pages 936–945. Pmlr, 2021.

[37] Luigi Bonati, GiovanniMaria Piccini, and Michele Parrinello. Deep learning the slow modes for rare events sampling. *Proceedings of the National Academy of Sciences*, 118 (44):e2113533118, 2021. doi: 10.1073/pnas.2113533118. URL https://www.pnas.org/doi/abs/10.1073/pnas.2113533118.

[38] Taha Boussaid, François Rousset, Vasile-Marian Scuturici, and Marc Clausse. Evaluation of Graph Neural Networks as Surrogate Model for District Heating Networks Simulation. In *36th International Conference on Efficiency, Cost, Optimization, Simulation and Environmental Impact of Energy Systems (ECOS 2023)*, pages 3182–3193, Las Palmas De Gran Canaria, Spain, June 2023. ECOS 2023. doi: 10.52202/069564-0286. URL https://hal.science/hal-04228699.

[39] Andrew Boutros, Eriko Nurvitadhi, Rui Ma, Sergey Gribok, Zhipeng Zhao, James C Hoe, Vaughn Betz, and Martin Langhammer. Beyond peak performance: Comparing the real performance of ai-optimized fpgas and gpus. In *2020 international conference on field-programmable technology (ICFPT)*, pages 10–19. IEEE, 2020.

[40] Gilles Brassard and Peter Hoyer. An exact quantum polynomial-time algorithm for simon's problem. In *Proceedings of the Fifth Israeli Symposium on Theory of Computing and Systems*, pages 12–23. IEEE, 1997.

[41] Randy Brown. Calendar queues: a fast O(1) priority queue implementation for the simulation event set problem. *Communications of the ACM*, 31:1220–1227, 1988. ISSN 0001-0782.

[42] Caterina Buizza, César Quilodrán Casas, Philip Nadler, Julian Mack, Stefano Marrone, Zainab Titus, Clémence Le Cornec, Evelyn Heylen, Tolga Dur, Luis Baca Ruiz, et al. Data learning: Integrating data assimilation and machine learning. *Journal of Computational Science*, 58:101525, 2022.

[43] Chengkun Cai, Xu Zhao, Haoliang Liu, Zhongyu Jiang, Tianfang Zhang, Zongkai Wu, Jenq-Neng Hwang, and Lei Li. The role of deductive and inductive reasoning in large language models. *arXiv preprint arXiv:2410.02892*, 2024.

[44] Wentong Cai, Philipp Andelfinger, Luca Bortolussi, Christopher Carothers, Dong Kevin Jin, Till Köster, Michael Lees, Jason Liu, Margaret Loper, Alessandro Pellegrini, et al. 4 working groups 4.1 intelligent modeling and simulation lifecycle. *Computer Science Methods for Effective and Sustainable Simulation Studies*, page 44, 2023.

[45] Diego Calanzone, Stefano Teso, and Antonio Vergari. Logically consistent language models via neuro-symbolic integration. In *The First Workshop on System-2 Reasoning at Scale, NeurIPS'24*.

[46] Donna Calhoun, Roy Overstreet, et al. Sensitivity analysis of a dynamical system using c++. *Scientific Programming*, 2(4):157–169, 1993.

[47] Christopher D. Carothers, Kalyan S. Perumalla, and Richard M. Fujimoto. Efficient optimistic parallel simulations using reverse computation. *ACM Trans. Model. Comput. Simul.*, 9(3):224–253, July 1999. ISSN 1049-3301. doi: 10.1145/347823.347828. URL `https://doi.org/10.1145/347823.347828`.

[48] Juan Carrasquilla, Di Luo, Felipe Pérez, Ashley Milsted, Bryan K Clark, Maksims Volkovs, and Leandro Aolita. Probabilistic simulation of quantum circuits using a deep-learning architecture. *Physical Review A*, 104(3):032610, 2021.

[49] Yolanda Carson and Anu Maria. Simulation optimization: methods and applications. In *Proceedings of the 29th Conference on Winter Simulation*, WSC '97, page 118–126, USA, 1997. IEEE Computer Society. ISBN 078034278X. doi: 10.1145/268437.268460. URL `https://doi.org/10.1145/268437.268460`.

[50] Lorenzo Casalino, Abigail C Dommer, Zied Gaieb, Emilia P Barros, Terra Sztain, Surl-Hee Ahn, Anda Trifan, Alexander Brace, Anthony T Bogetti, Austin Clyde, et al. Ai-driven multiscale simulations illuminate mechanisms of sars-cov-2 spike dynamics. *The International Journal of High Performance Computing Applications*, 35(5):432–451, 2021.

[51] Nurcin Celik, Seungho Lee, Karthik Vasudevan, and Young-Jun Son. Dddas-based multi-fidelity simulation framework for supply chain systems. *IIE transactions*, 42(5):325–341, 2010.

[52] Kanianthra Mani Chandy and Jaydev Misra. Distributed simulation: A case study in design and verification of distributed programs. *IEEE Transactions on Software Engineering*, SE-5(5):440–452, September 1979. ISSN 0098-5589,1939-3520. doi: 10.1109/tse.1979.230182.

[53] Tanmoy Chatterjee, Souvik Chakraborty, and Rajib Chowdhury. A critical review of surrogate assisted robust design optimization. *Archives of Computational Methods in Engineering*, 26(1):245–274, Jan 2019. ISSN 1886-1784. doi: 10.1007/s11831-017-9240-5. URL `https://doi.org/10.1007/s11831-017-9240-5`.

[54] Swarat Chaudhuri and Armando Solar-Lezama. Smooth interpretation. *ACM Sigplan Notices*, 45(6):279–291, 2010.

[55] Swarat Chaudhuri, Kevin Ellis, Oleksandr Polozov, Rishabh Singh, Armando Solar-Lezama, and Yisong Yue. Neurosymbolic programming. *Foundations and Trends® in Programming Languages*, 7(3):158–243, 2021. ISSN 2325-1107. doi: 10.1561/2500000049. URL `http://dx.doi.org/10.1561/2500000049`.

[56] Yiran Chen, Yuan Xie, Linghao Song, Fan Chen, and Tianqi Tang. A survey of accelerator architectures for deep neural networks. *Engineering*, 6(3):264–274, 2020.

[57] Sibo Cheng, Cesar Quilodran-Casas, Said Ouala, Alban Farchi, Che Liu, Pierre Tandeo, Ronan Fablet, Didier Lucor, Bertrand Iooss, Julien Brajard, Dunhui Xiao, Tijana Janjic, Weiping Ding, Yike Guo, Alberto Carrassi, Marc Bocquet, and Rossella Arcucci. Machine learning with data assimilation and uncertainty quantification for dynamical systems: a review, 2023. URL `https://arxiv.org/abs/2303.10462`.

[58] Sibo Cheng, César Quilodrán-Casas, Said Ouala, Alban Farchi, Che Liu, Pierre Tandeo, Ronan Fablet, Didier Lucor, Bertrand Iooss, Julien Brajard, et al. Machine learning with data assimilation and uncertainty quantification for dynamical systems: a review. *IEEE/CAA Journal of Automatica Sinica*, 10(6):1361–1387, 2023.

[59] Stephen E Chick. Bayesian analysis for simulation input and output. In *Proceedings of the 29th conference on Winter simulation*, pages 253–260, 1997.

[60] Seon Han Choi, Kyung-Min Seo, and Tag Gon Kim. Accelerated simulation of discrete event dynamic systems via a multi-fidelity modeling framework. *Applied Sciences*, 7(10): 1056, 2017.

[61] Ayush Chopra, Esma Gel, Jayakumar Subramanian, Balaji Krishnamurthy, Santiago Romero-Brufau, Kalyan S. Pasupathy, Thomas C. Kingsley, and Ramesh Raskar. Deepabm: Scalable, efficient and differentiable agent-based simulations via graph neural networks, 2021. URL `https://arxiv.org/abs/2110.04421`.

[62] Minghan Chu and Weicheng Qian. A deep learning approach for epistemic uncertainty quantification of turbulent flow simulations. *arXiv preprint arXiv:2405.08148*, 2024.

[63] Eric Chung, Jeremy Fowers, Kalin Ovtcharov, Michael Papamichael, Adrian Caulfield, Todd Massengill, Ming Liu, Daniel Lo, Shlomi Alkalay, Michael Haselman, et al. Serving dnns in real time at datacenter scale with project brainwave. *iEEE Micro*, 38(2):8–20, 2018.

[64] Eric Chung, Yalchin Efendiev, Wing Tat Leung, Sai-Mang Pun, and Zecheng Zhang. Multi-agent reinforcement learning accelerated mcmc on multiscale inversion problem. *arXiv preprint arXiv:2011.08954*, 2020.

[65] Edmund M Clarke. Model checking. In *Foundations of Software Technology and Theoretical Computer Science: 17th Conference Kharagpur, India, December 18–20, 1997 Proceedings 17*, pages 54–56. Springer, 1997.

[66] Robert Clay, Jonathan A Ward, Patricia Ternes, Le-Minh Kieu, and Nick Malleson. Real-time agent-based crowd simulation with the reversible jump unscented kalman filter. *Simulation Modelling Practice and Theory*, 113:102386, 2021.

[67] Paolo Conti, Mengwu Guo, Andrea Manzoni, and Jan S. Hesthaven. Multi-fidelity surrogate modeling using long short-term memory networks. *Computer Methods in Applied Mechanics and Engineering*, 404:115811, 2023. ISSN 0045-7825. doi: https://doi.org/10.1016/j.cma.2022.115811. URL `https://www.sciencedirect.com/science/article/pii/S0045782522007678`.

[68] George Corliss, Andreas Griewank, S Wright, and T Robey. Automatic differentiation applied to unsaturated flow-adol-c case study. Technical report, Argonne National Lab., IL (United States). Mathematics and Computer Science Div., 1992.

[69] Biagio Cosenza, Nikita Popov, Ben Juurlink, Paul Richmond, Mozhgan Kabiri Chimeh, Carmine Spagnuolo, Gennaro Cordasco, and Vittorio Scarano. Openabl: a domain-specific language for parallel and distributed agent-based simulations. In *Euro-Par 2018: Parallel Processing: 24th International Conference on Parallel and Distributed Computing, Turin, Italy, August 27-31, 2018, Proceedings 24*, pages 505–518. Springer, 2018.

[70] Patrick Cousot. Abstract interpretation. *ACM Computing Surveys (CSUR)*, 28(2):324–328, 1996.

[71] Kyle Cranmer, Johann Brehmer, and Gilles Louppe. The frontier of simulation-based inference. *Proceedings of the National Academy of Sciences*, 117(48):30055–30062, 2020. doi: 10.1073/pnas.1912789117. URL https://www.pnas.org/doi/abs/10.1073/pnas.1912789117.

[72] Miles Cranmer, Alvaro Sanchez Gonzalez, Peter Battaglia, Rui Xu, Kyle Cranmer, David Spergel, and Shirley Ho. Discovering symbolic models from deep learning with inductive biases. *Advances in neural information processing systems*, 33:17429–17442, 2020.

[73] K. Crombecq, E. Laermans, and T. Dhaene. Efficient space-filling and non-collapsing sequential design strategies for simulation-based modeling. *European Journal of Operational Research*, 214(3):683–696, 2011. ISSN 0377-2217. doi: https://doi.org/10.1016/j.ejor.2011.05.032. URL https://www.sciencedirect.com/science/article/pii/S0377221711004577.

[74] Karel Crombecq, Dirk Gorissen, Dirk Deschrijver, and Tom Dhaene. A novel hybrid sequential design strategy for global surrogate modeling of computer experiments. *SIAM Journal on Scientific Computing*, 33(4):1948–1974, 2011. doi: 10.1137/090761811. URL https://doi.org/10.1137/090761811.

[75] Chunfeng Cui, Cole Hawkins, and Zheng Zhang. Tensor methods for generating compact uncertainty quantification and deep learning models. In *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 1–6. IEEE, 2019.

[76] Jingjing Cui, Philippe JS de Brouwer, Steven Herbert, Philip Intallura, Cahit Kargi, Georgios Korpas, Alexandre Krajenbrink, William Shooshmith, Ifan Williams, and Ban Zheng. Quantum monte carlo integration for simulation-based optimisation. *arXiv preprint arXiv:2410.03926*, 2024.

[77] Judith S Dahmann, Richard M Fujimoto, and Richard M Weatherly. The department of defense high level architecture. In *Proceedings of the 29th conference on Winter simulation*, pages 142–149, 1997.

[78] Avishek Das, Dominic C Rose, Juan P Garrahan, and David T Limmer. Reinforcement learning of rare diffusive dynamics. *The Journal of Chemical Physics*, 155(13), 2021.

[79] Stéphane d'Ascoli, Sören Becker, Alexander Mathis, Philippe Schwaller, and Niki Kilbertus. Odeformer: Symbolic regression of dynamical systems with transformers, 2023. URL https://arxiv.org/abs/2310.05573.

[80] Mike Davies, Narayan Srinivasa, Tsung-Han Lin, Gautham Chinya, Yongqiang Cao, Sri Harsha Choday, Georgios Dimou, Prasad Joshi, Nabil Imam, Shweta Jain, Yuyun Liao, Chit-Kwan Lin, Andrew Lines, Ruokun Liu, Deepak Mathaikutty, Steven McCoy, Arnab Paul, Jonathan Tse, Guruguhanathan Venkataramanan, Yi-Hsin Weng, Andreas Wild, Yoonseok Yang, and Hong Wang. Loihi: A neuromorphic manycore processor with on-chip learning. *IEEE micro*, 38(1):82–99, January 2018. ISSN 0272-1732,1937-4143. doi: 10.1109/mm.2018.112130359.

[81] Jie Ding, Vahid Tarokh, and Yuhong Yang. Model selection techniques: An overview. *IEEE Signal Processing Magazine*, 35(6):16–34, 2018.

[82] Lawrence Charles Ward Dixon, P Dolan, and RC Price. *Finite element optimisation: The use of structured automatic differentiation*. Hatfield Polytechnic, Numerical Optimisation Centre, 1986.

[83] Yihong Dong, Ge Li, and Zhi Jin. Codep: grammatical seq2seq model for general-purpose code generation. In *Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis*, pages 188–198, 2023.

[84] Vedran Dunjko and Hans J Briegel. Machine learning & artificial intelligence in the quantum domain: a review of recent progress. *Reports on Progress in Physics*, 81(7): 074001, 2018.

[85] Roland Ewald. *Automatic algorithm selection for complex simulation problems*. Springer, 2012.

[86] Yuping Fan, Zhiling Lan, Taylor Childers, Paul Rich, William Allcock, and Michael E. Papka. Deep reinforcement agent for scheduling in hpc. In *2021 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 807–816, 2021. doi: 10.1109/IPDPS49936.2021.00090.

[87] Alban Farchi, Marc Bocquet, Patrick Laloyaux, Massimo Bonavita, and Quentin Malartic. A comparison of combined data assimilation and machine learning methods for offline and online model error correction. *Journal of computational science*, 55:101468, 2021.

[88] Niclas Feldkamp and Steffen Strassburger. From explainable ai to explainable simulation: Using machine learning and xai to understand system robustness. In *Proceedings of the 2023 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*, SIGSIM-PADS '23, page 96–106, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400700309. doi: 10.1145/3573900.3591114. URL `https://doi.org/10.1145/3573900.3591114`.

[89] Niclas Feldkamp, Soeren Bergmann, Florian Conrad, and Steffen Strassburger. A method using generative adversarial networks for robustness optimization. *ACM Trans. Model. Comput. Simul.*, 32(2), mar 2022. ISSN 1049-3301. doi: 10.1145/3503511. URL `https://doi.org/10.1145/3503511`.

[90] Andreas K Fidjeland, Etienne B Roesch, Murray P Shanahan, and Wayne Luk. NeMo: A platform for neural modelling of spiking neurons using GPUs. In *Proceedings of the 20th IEEE International Conference on Application-specific Systems, Architectures and Processors*, ASAP, pages 137–144, Piscataway, NJ, USA, July 2009. IEEE. ISBN 9780769537320. doi: 10.1109/ASAP.2009.24.

[91] C Daniel Freeman, Erik Frey, Anton Raichuk, Sertan Girgin, Igor Mordatch, and Olivier Bachem. Brax–a differentiable physics engine for large scale rigid body simulation. *arXiv preprint arXiv:2106.13281*, 2021.

[92] Jonas Friederich, Wentong Cai, Boon Ping Gan, and Sanja Lazarova-Molnar. Equipment-centric data-driven reliability assessment of complex manufacturing systems. In *Proceedings of the 2023 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*, pages 62–72, 2023.

[93] Erika Frydenlund, Joseph Martínez, Jose J Padilla, Katherine Palacio, and David Shuttleworth. Modeler in a box: how can large language models aid in the simulation modeling process? *Simulation*, page 00375497241239360, 2024.

[94] Richard M Fujimoto. Performance of time warp under synthetic workloads. In David Nicol, editor, *Distributed Simulation*, PADS '90, pages 23–28, San Diego, CA, USA, 1990. Society for Computer Simulation International.

[95] Richard M Fujimoto. Research challenges in parallel and distributed simulation. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 26(4):1–29, 2016.

[96] Richard M Fujimoto. Parallel simulation of discrete systems. *Parallel Processing in Computational Mechanics*, pages 151–182, 2020.

[97] Steve B Furber, David R Lester, Luis A Plana, Jim D Garside, Eustace Painkras, Steve Temple, and Andrew D Brown. Overview of the spinnaker system architecture. *IEEE transactions on computers*, 62(12):2454–2467, 2012.

[98] Julien Gacon, Christa Zoufal, and Stefan Woerner. Quantum-enhanced simulation-based optimization. In *2020 IEEE International Conference on Quantum Computing and Engineering (QCE)*, pages 47–55. IEEE, 2020.

[99] Artur d'Avila Garcez and Luis C Lamb. Neurosymbolic ai: The 3 rd wave. *Artificial Intelligence Review*, 56(11):12387–12406, 2023.

[100] Tomas Geffner, Javier Antoran, Adam Foster, Wenbo Gong, Chao Ma, Emre Kiciman, Amit Sharma, Angus Lamb, Martin Kukla, Nick Pawlowski, et al. Deep end-to-end causal inference. *arXiv preprint arXiv:2202.02195*, 2022.

[101] Amir Ghorbani, Vahid Ghorbani, Morteza Nazari-Heris, and Somayeh Asadi. Data assimilation for agent-based models. *Mathematics*, 11(20):4296, 2023.

[102] Lachlan Gibson, Marcus Hoerger, and Dirk Kroese. A flow-based generative model for rare-event simulation, 2023. URL `https://arxiv.org/abs/2305.07863`.

[103] Daniel T Gillespie. Approximate accelerated stochastic simulation of chemically reacting systems. *The Journal of chemical physics*, 115(4):1716–1733, 2001.

[104] Manuel Gloeckler, Michael Deistler, Christian Weilbach, Frank Wood, and Jakob H Macke. All-in-one simulation-based inference. *arXiv preprint arXiv:2404.09636*, 2024.

[105] Ángel Goñi Moreno. Biocomputation: Moving beyond turing with living cellular computers. *Commun. ACM*, 67(6):70–77, may 2024. ISSN 0001-0782. doi: 10.1145/3635470. URL `https://doi.org/10.1145/3635470`.

[106] Wei-Bo Gong and Yu-Chi Ho. Smoothed (conditional) perturbation analysis of discrete event dynamical systems. *IEEE Transactions on Automatic Control*, 32(10):858–866, 1987.

[107] Paloma Gonzalez-Rojas, Andrew Emmel, Luis Martinez, Neil Malur, and Gregory Rutledge. Augmenting control over exploration space in molecular dynamics simulators to streamline de novo analysis through generative control policies. *arXiv preprint arXiv:2306.14705*, 2023.

[108] Paula Gradu, John Hallman, Daniel Suo, Alex Yu, Naman Agarwal, Udaya Ghai, Karan Singh, Cyril Zhang, Anirudha Majumdar, and Elad Hazan. Deluca–a differentiable control library: Environments, methods, and benchmarking. *arXiv preprint arXiv:2102.09968*, 2021.

[109] Matthew Graham and Amos Storkey. Asymptotically exact inference in differentiable generative models. In *Artificial Intelligence and Statistics*, pages 499–508. PMLR, 2017.

[110] Joe G Greener. Differentiable simulation to develop molecular dynamics force fields for disordered proteins. *bioRxiv*, 2024. doi: 10.1101/2023.08.29.555352. URL `https://www.biorxiv.org/content/early/2024/01/15/2023.08.29.555352`.

[111] Andreas Griewank and Andrea Walther. Introduction to automatic differentiation. In *PAMM: Proceedings in Applied Mathematics and Mechanics*, volume 2, pages 45–49. Wiley Online Library, 2003.

[112] Michael Grohs, Luka Abb, Nourhan Elsayed, and Jana-Rebecca Rehse. Large language models can accomplish business process management tasks. In *International Conference on Business Process Management*, pages 453–465. Springer, 2023.

[113] Gerrit Großmann, Julian Zimmerlin, Michael Backenköhler, and Verena Wolf. Unsupervised relational inference using masked reconstruction. *Applied Network Science*, 8(1):18, 2023.

[114] L. Grozinger, M. Amos, and T. E. Gorochowski et al. Pathways to cellular supremacy in biocomputing. *Nat. Commun*, 10(5250), 2019. doi: https://doi.org/10.1038/s41467-019-13232-z.

[115] Sumit Gulwani, Oleksandr Polozov, and Rishabh Singh. Program synthesis. *Foundations and Trends® in Programming Languages*, 4(1-2):1–119, 2017. ISSN 2325-1107. doi: 10.1561/2500000010. URL `http://dx.doi.org/10.1561/2500000010`.

[116] Rohan Gupta, Devesh Srivastava, Mehar Sahu, Swati Tiwari, Rashmi K Ambasta, and Pravir Kumar. Artificial intelligence to deep learning: machine intelligence approach for drug discovery. *Molecular diversity*, 25:1315–1360, 2021.

[117] David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018.

[118] Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. *arXiv preprint arXiv:2010.02193*, 2020.

[119] Leonard A Harris, Justin S Hogg, José-Juan Tapia, John AP Sekar, Sanjana Gupta, Ilya Korsunsky, Arshi Arora, Dipak Barua, Robert P Sheehan, and James R Faeder. Bionetgen 2.2: advances in rule-based modeling. *Bioinformatics*, 32(21):3366–3368, 2016.

[120] Xin He, Kaiyong Zhao, and Xiaowen Chu. Automl: A survey of the state-of-the-art. *Knowledge-based systems*, 212:106622, 2021.

[121] Eric Heiden, Miles Macklin, Yashraj Narang, Dieter Fox, Animesh Garg, and Fabio Ramos. Disect: A differentiable simulation engine for autonomous robotic cutting. *arXiv preprint arXiv:2105.12244*, 2021.

[122] Tobias Helms, Roland Ewald, Stefan Rybacki, and Adelinde M Uhrmacher. Automatic runtime adaptation for component-based simulation algorithms. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 26(1):1–24, 2015.

[123] Yu-Chi Ho and Christos Cassandras. A new approach to the analysis of discrete event dynamic systems. *Automatica*, 19(2):149–167, 1983.

[124] Gary E Horne and Theodore E Meyer. Data farming: Discovering surprise. In *Proceedings of the 2004 Winter Simulation Conference, 2004.*, volume 1. IEEE, 2004.

[125] Hector J Hortua, Riccardo Volpi, Dimitri Marinelli, and Luigi Malago. Accelerating mcmc algorithms through bayesian deep networks. *arXiv preprint arXiv:2011.14276*, 2020.

[126] Chun Kit Jeffery Hou and Kamran Behdinan. Dimensionality reduction in surrogate modeling: A review of combined methods. *Data Science and Engineering*, 7(4):402–427, Dec 2022. ISSN 2364-1541. doi: 10.1007/s41019-022-00193-5. URL `https://doi.org/10.1007/s41019-022-00193-5`.

[127] Taylor A Howell, Simon Le Cleac'h, J Zico Kolter, Mac Schwager, and Zachary Manchester. Dojo: A differentiable simulator for robotics. *arXiv preprint arXiv:2203.00806*, 9, 2022.

[128] Xiaolin Hu. *Dynamic data-driven simulation: real-time data for dynamic system analysis and prediction*. World Scientific, 2023.

[129] Xiaolin Hu and Peisheng Wu. A data assimilation framework for discrete event simulations. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 29(3): 1–26, 2019.

[130] Yuanming Hu, Jiancheng Liu, Andrew Spielberg, Joshua B Tenenbaum, William T Freeman, Jiajun Wu, Daniela Rus, and Wojciech Matusik. Chainqueen: A real-time differentiable physical simulator for soft robotics. In *2019 International conference on robotics and automation (ICRA)*, pages 6265–6271. IEEE, 2019.

[131] Xinru Hua, Rasool Ahmad, Jose Blanchet, and Wei Cai. Accelerated sampling of rare events using a neural network bias potential, 2024. URL `https://arxiv.org/abs/2401.06936`.

[132] Zhiao Huang, Yuanming Hu, Tao Du, Siyuan Zhou, Hao Su, Joshua B Tenenbaum, and Chuang Gan. Plasticinelab: A soft-body manipulation benchmark with differentiable physics. *arXiv preprint arXiv:2104.03311*, 2021.

[133] Zhiyuan Huang, Mansur Arief, Henry Lam, and Ding Zhao. Evaluation uncertainty in data-driven self-driving testing. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 1902–1907. IEEE, 2019.

[134] Kelli D Humbird and Jayson Luc Peterson. Transfer learning driven design optimization for inertial confinement fusion. *Physics of Plasmas*, 29(10), 2022.

[135] Mauro Ianni, Romolo Marotta, Davide Cingolani, Alessandro Pellegrini, and Francesco Quaglia. The ultimate share-everything PDES system. In *Proceedings of the 2018 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*, SIGSIM-PADS '18, pages 73–84, New York, NY, USA, May 2018. ACM. ISBN 9781450350921. doi: 10.1145/3200921.3200931.

[136] Martin Ihrig. A new research architecture for the simulation era. *Seminal Contributions to Modelling and Simulation: 30 Years of the European Council of Modelling and Simulation*, pages 47–55, 2016.

[137] John Ingraham, Adam Riesselman, Chris Sander, and Debora Marks. Learning protein structure with a differentiable simulator. In *International conference on learning representations*, 2018.

[138] Ilya Jackson, Maria Jesus Saenz, and Dmitry Ivanov. From natural language to simulations: applying ai to automate simulation modelling of logistics systems. *International Journal of Production Research*, 62(4):1434–1457, 2024.

[139] David R Jefferson. Virtual time. *ACM Transactions on Programming Languages and Systems*, 7(3):404–425, July 1985. ISSN 0164-0925. doi: 10.1145/3916.3988.

[140] Yen Jeng and Chou Chien-Chun. Automatic generation and numerical integration of differential-algebraic equations of multibody dynamics. *Computer methods in applied mechanics and engineering*, 104(3):317–331, 1993.

[141] D Jenkins and G Peterson. Gpu accelerated stochastic simulation. In *Symposium on Application Accelerators in High Performance Computing (SAAHPC)*. Citeseer, 2010.

[142] Xin-fang Ji, Yong Zhang, Chun-lin He, Jin-xin Cheng, Dun-wei Gong, Xiao-zhi Gao, and Yi-nan Guo. Surrogate and autoencoder-assisted multitask particle swarm optimization for high-dimensional expensive multimodal problems. *IEEE Transactions on Evolutionary Computation*, 2023.

[143] Bai Jiang, Tung-yu Wu, Charles Zheng, and Wing H Wong. Learning summary statistic for approximate bayesian computation via deep neural network. *Statistica Sinica*, pages 1595–1618, 2017.

[144] Jinhao Jiang, Kun Zhou, Zican Dong, Keming Ye, Wayne Xin Zhao, and Ji-Rong Wen. Structgpt: A general framework for large language model to reason over structured data. *arXiv preprint arXiv:2305.09645*, 2023.

[145] Juyong Jiang, Fan Wang, Jiasi Shen, Sungju Kim, and Sunghun Kim. A survey on large language models for code generation. *arXiv preprint arXiv:2406.00515*, 2024.

[146] Larry Jin, Hannah Lu, and Gege Wen. Fast uncertainty quantification of reservoir simulation with variational u-net, 2019. URL `https://arxiv.org/abs/1907.00718`.

[147] Norm Jouppi, George Kurian, Sheng Li, Peter Ma, Rahul Nagarajan, Lifeng Nai, Nishant Patil, Suvinay Subramanian, Andy Swing, Brian Towles, et al. Tpu v4: An optically reconfigurable supercomputer for machine learning with hardware support for embeddings. In *Proceedings of the 50th Annual International Symposium on Computer Architecture*, pages 1–14, 2023.

[148] Richard Jozsa. On the simulation of quantum circuits. *arXiv preprint quant-ph/0603163*, 2006.

[149] Jakob Kallestad, Ramin Hasibi, Ahmad Hemmati, and Kenneth Sörensen. A general deep reinforcement learning hyperheuristic framework for solving combinatorial optimization problems. *European Journal of Operational Research*, 309(1):446–468, 2023.

[150] Pierre-Alexandre Kamienny, Stéphane d'Ascoli, Guillaume Lample, and François Charton. End-to-end symbolic regression with transformers. *Advances in Neural Information Processing Systems*, 35:10269–10281, 2022.

[151] George Em Karniadakis, Ioannis G. Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440, Jun 2021. ISSN 2522-5820. doi: 10.1038/s42254-021-00314-5. URL `https://doi.org/10.1038/s42254-021-00314-5`.

[152] Sarah M Keating, Dagmar Waltemath, Matthias König, Fengkai Zhang, Andreas Dräger, Claudine Chaouiya, Frank T Bergmann, Andrew Finney, Colin S Gillespie, Tomáš Helikar, et al. Sbml level 3: an extensible format for the exchange and reuse of biological models. *Molecular systems biology*, 16(8):e9110, 2020.

[153] Marc C Kennedy and Anthony O'Hagan. Bayesian calibration of computer models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(3):425–464, 2001.

[154] Pascal Kerschke, Holger H Hoos, Frank Neumann, and Heike Trautmann. Automated algorithm selection: Survey and perspectives. *Evolutionary computation*, 27(1):3–45, 2019.

[155] Hassan Khosravi and Bahareh Bina. A survey on statistical relational learning. In Atefeh Farzindar and Vlado Kešelj, editors, *Advances in Artificial Intelligence*, pages 256–268, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg. ISBN 978-3-642-13059-5.

[156] Thomas Kipf, Ethan Fetaya, Kuan-Chieh Wang, Max Welling, and Richard Zemel. Neural relational inference for interacting systems. In *International conference on machine learning*, pages 2688–2697. PMLR, 2018.

[157] Mariam Kiran, Paul Richmond, Mike Holcombe, Lee Shawn Chin, David Worth, and Chris Greenough. Flame: simulating large populations of agents on parallel hardware architectures. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1*, pages 1633–1636, 2010.

[158] James C Knight, Anton Komissarov, and Thomas Nowotny. Pygenn: a python library for gpu-enhanced neural networks. *Frontiers in Neuroinformatics*, 15:659005, 2021.

[159] Till Köster, Leon Herrmann, Philipp Andelfinger, and Adelinde Uhrmacher. Gpu-accelerated simulation ensembles of stochastic reaction networks. In *2022 Winter Simulation Conference (WSC)*, pages 2570–2581. IEEE, 2022.

[160] Justin N. Kreikemeyer and Philipp Andelfinger. Smoothing Methods for Automatic Differentiation Across Conditional Branches. *IEEE Access*, 11:143190–143211, 2023.

[161] Justin N Kreikemeyer, Kevin Burrage, and Adelinde M Uhrmacher. Discovering biochemical reaction models by evolving libraries. In *International Conference on Computational Methods in Systems Biology*, pages 117–136. Springer, 2024.

[162] D. Kudithipudi, C. Schuman, C.M. Vineyard, et al. Neuromorphic computing at scale. *Nature*, 637:801–812, 2025. doi: 10.1038/s41586-024-08253-8.

[163] Georg Kunz, Daniel Schemmel, James Gross, and Klaus Wehrle. Multi-level parallelism for time-and cost-efficient parallel discrete event simulation on gpus. In *2012 ACM/IEEE/SCS 26th Workshop on Principles of Advanced and Distributed Simulation*, pages 23–32. IEEE, 2012.

[164] Bogdan Kustowski, Jim A Gaffney, Brian K Spears, Gemma J Anderson, Rushil Anirudh, Peer-Timo Bremer, Jayaraman J Thiagarajan, Michael KG Kruse, and Ryan C Nora. Suppressing simulation bias in multi-modal data using transfer learning. *Machine Learning: Science and Technology*, 3(1):015035, 2022.

[165] Alexander Lavin, David Krakauer, Hector Zenil, Justin Gottschlich, Tim Mattson, Johann Brehmer, Anima Anandkumar, Sanjay Choudry, Kamil Rocki, Atılım Güneş Baydin, Carina Prunkl, Brooks Paige, Olexandr Isayev, Erik Peterson, Peter L McMahon, Jakob Macke, Kyle Cranmer, Jiaxin Zhang, Haruko Wainwright, Adi Hanuka, Manuela Veloso, Samuel Assefa, Stephan Zheng, and Avi Pfeffer. Simulation intelligence: Towards a new generation of scientific methods. *arXiv*, (2112.03235), December 2021.

[166] Averill M. Law. Statistical analysis of simulation output data: The practical state of the art. In *2020 Winter Simulation Conference (WSC)*, pages 1117–1127, 2020. doi: 10.1109/WSC48552.2020.9383993.

[167] Pierre L'Ecuyer, Michel Mandjes, and Bruno Tuffin. *Importance Sampling in Rare Event Simulation*, chapter 2, pages 17–38. John Wiley & Sons, Ltd, 2009. ISBN 9780470745403. doi: https://doi.org/10.1002/9780470745403.ch2. URL `https://onlinelibrary.wiley.com/doi/abs/10.1002/9780470745403.ch2`.

[168] Lawrence M Leemis and Stephen K Park. *Discrete-Event Simulation: A First Course*. Pearson, Upper Saddle River, NJ, USA, 1 edition, January 2006.

[169] Madlene Leißau and Christoph Laroque. Reverse engineering the future–an automated backward simulation approach to on-time production in the semiconductor industry. In *2023 Winter Simulation Conference (WSC)*, pages 2040–2051. IEEE, 2023.

[170] Jun-Xue Leng, Yuan Feng, Wei Huang, Yang Shen, and Zhen-Guo Wang. Variable-fidelity surrogate model based on transfer learning and its application in multidisciplinary design optimization of aircraft. *Physics of Fluids*, 36(1):017131, 01 2024. ISSN 1070-6631. doi: 10.1063/5.0188386. URL `https://doi.org/10.1063/5.0188386`.

[171] Dongrui Li and Jinghui Zhong. Dimensionally aware multi-objective genetic programming for automatic crowd behavior modeling. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 30(3):1–24, 2020.

[172] Jiachen Li, Chuanbo Hua, Hengbo Ma, Jinkyoo Park, Victoria Dax, and Mykel J Kochenderfer. Multi-agent dynamic relational reasoning for social robot navigation. *arXiv preprint arXiv:2401.12275*, 2024.

[173] Jian-Yu Li, Zhi-Hui Zhan, Hua Wang, and Jun Zhang. Data-driven evolutionary algorithm with perturbation-based ensemble surrogates. *IEEE Transactions on Cybernetics*, 51(8):3925–3937, 2020.

[174] Zhongwei Lin, Carl Tropper, Yiping Yao, Robert A Mcdougal, Mohammand Nazrul Ishlam Patoary, William W Lytton, and Michael L Hines. Load balancing for multi-threaded pdes of stochastic reaction-diffusion in neurons. *Journal of Simulation*, 11(3):267–284, 2017.

[175] Zhan Ling, Yunhao Fang, Xuanlin Li, Zhiao Huang, Mingu Lee, Roland Memisevic, and Hao Su. Deductive verification of chain-of-thought reasoning. *Advances in Neural Information Processing Systems*, 36, 2024.

[176] Hanmeng Liu, Ruoxi Ning, Zhiyang Teng, Jian Liu, Qiji Zhou, and Yue Zhang. Evaluating the logical reasoning ability of chatgpt and gpt-4. *arXiv preprint arXiv:2304.03439*, 2023.

[177] Qi Liu, Yuanqi Du, Fan Feng, Qiwei Ye, and Jie Fu. Structural causal model for molecular dynamics simulation. In *NeurIPS 2022 AI for Science: Progress and Promises*.

[178] Tianyi Liu, Yifan Lin, and Enlu Zhou. Bayesian stochastic gradient descent for stochastic optimization with streaming input data. *SIAM Journal on Optimization*, 34(1):389–418, 2024.

[179] Xinhu Liu and Philipp Andelfinger. Time warp on the GPU: Design and assessment. In *Proceedings of the 2017 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*, SIGSIM-PADS '17, pages 109–120, New York, NY, USA, May 2017. ACM. ISBN 9781450344890. doi: 10.1145/3064911.3064912.

[180] Lennart Ljung. Perspectives on system identification. *Annual Reviews in Control*, 34(1): 1–12, 2010.

[181] Denghui Lu, Han Wang, Mohan Chen, Lin Lin, Roberto Car, E Weinan, Weile Jia, and Linfeng Zhang. 86 pflops deep potential molecular dynamics simulation of 100 million atoms with ab initio accuracy. *Computer Physics Communications*, 259:107624, 2021.

[182] Jiali Luan and Zheng Zhang. Prediction of multidimensional spatial variation data via bayesian tensor completion. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 39(2):547–551, 2019.

[183] Jan-Matthis Lueckmann, Pedro J Goncalves, Giacomo Bassetto, Kaan Öcal, Marcel Nonnenmacher, and Jakob H Macke. Flexible statistical inference for mechanistic models of neural dynamics. *Advances in neural information processing systems*, 30, 2017.

[184] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL `https://proceedings.neurips.cc/paper_files/paper/2017/file/8a20a8621978632d76c43dfd28b67767-Paper.pdf`.

[185] Kazuhiro Maeda and Hiroyuki Kurata. Automatic generation of sbml kinetic models from natural language texts using gpt. *International Journal of Molecular Sciences*, 24 (8):7296, 2023.

[186] Nour Makke and Sanjay Chawla. Interpretable scientific discovery with symbolic regression: a review. *Artificial Intelligence Review*, 57(1):2, 2024.

[187] Nick Malleson, Kevin Minors, Le-Minh Kieu, Jonathan A Ward, Andrew A West, and Alison Heppenstall. Simulating crowds in real time with agent-based modelling and a particle filter. *arXiv preprint arXiv:1909.09397*, 2019.

[188] Zohar Manna and Richard J. Waldinger. Toward automatic program synthesis. *Commun. ACM*, 14(3):151–165, mar 1971. ISSN 0001-0782. doi: 10.1145/362566.362568. URL `https://doi.org/10.1145/362566.362568`.

[189] Romolo Marotta and Alessandro Pellegrini. Model-driven engineering for high-performance parallel discrete event simulations on heterogeneous architectures. In H Lam, E Azar, D Batur, S Gao, W Xie, S R Hunter, and M D Rossetti, editors, *Proceedings of the 2024 Winter Simulation Conference*, WSC'24, Piscataway, NJ, USA, December 2024. IEEE.

[190] Romolo Marotta, Mauro Ianni, Alessandro Pellegrini, and Francesco Quaglia. A conflict-resilient lock-free calendar queue for scalable share-everything PDES platforms. In *Proceedings of the 2017 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*, SIGSIM-PADS '17, pages 15–26, New York, NY, USA, May 2017. Association for Computing Machinery. ISBN 9781450344890. doi: 10.1145/3064911.3064926.

[191] Julien Martinelli, Jeremy Grignard, Sylvain Soliman, Annabelle Ballesta, and François Fages. Reactmine: a statistical search algorithm for inferring chemical reactions from time series data. *arXiv preprint arXiv:2209.03185*, 2022.

[192] Nina Mazyavkina, Sergey Sviridov, Sergei Ivanov, and Evgeny Burnaev. Reinforcement learning for combinatorial optimization: A survey. *Computers & Operations Research*, 134:105400, 2021.

[193] Kazem Meidani, Parshin Shojaee, Chandan K. Reddy, and Amir Barati Farimani. Snip: Bridging mathematical symbolic and numeric realms with unified pre-training, 2024. URL `https://arxiv.org/abs/2310.02227`.

[194] Daniel Melcer, Nathan Fulton, Sanjay Krishna Gouda, and Haifeng Qian. Constrained decoding for code language models via efficient left and right quotienting of context-sensitive grammars. *arXiv preprint arXiv:2402.17988*, 2024.

[195] Valentyn Melnychuk, Dennis Frauen, and Stefan Feuerriegel. Causal transformer for estimating counterfactual outcomes. In *International Conference on Machine Learning*, pages 15293–15329. PMLR, 2022.

[196] Chuizheng Meng, Sungyong Seo, Defu Cao, Sam Griesemer, and Yan Liu. When physics meets machine learning: A survey of physics-informed machine learning, 2022. URL `https://arxiv.org/abs/2203.16797`.

[197] Sina Meraji, Wei Zhang, and Carl Tropper. A multi-state q-learning approach for the dynamic load balancing of time warp. In *2010 IEEE Workshop on Principles of Advanced and Distributed Simulation*, pages 1–8. IEEE, 2010.

[198] Diego Mesquita, Paul Blomstedt, and Samuel Kaski. Embarrassingly parallel mcmc using deep invertible transformations. In *Uncertainty in Artificial Intelligence*, pages 1244–1252. PMLR, 2020.

[199] Zhaobin Mo, Yongjie Fu, Daran Xu, and Xuan Di. Trafficflowgan: Physics-informed flow based generative adversarial network for uncertainty quantification. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 323–339. Springer, 2022.

[200] Dharmendra S. Modha, Filipp Akopyan, Alexander Andreopoulos, Rathinakumar Appuswamy, John V. Arthur, Andrew S. Cassidy, Pallab Datta, Michael V. DeBole, Steven K. Esser, Carlos Ortega Otero, Jun Sawada, Brian Taba, Arnon Amir, Deepika Bablani, Peter J. Carlson, Myron D. Flickner, Rajamohan Gandhasri, Guillaume J. Garreau, Megumi Ito, Jennifer L. Klamo, Jeffrey A. Kusnitz, Nathaniel J. McClatchey, Jeffrey L. McKinstry, Yutaka Nakamura, Tapan K. Nayak, William P. Risk, Kai Schleupen, Ben Shaw, Jay Sivagnaname, Daniel F. Smith, Ignacio Terrizzano, and Takanori Ueda. Neural inference at the frontier of energy, space, and time. *Science*, 382(6668):329–335, 2023. doi: 10.1126/science.adh1174. URL `https://www.science.org/doi/abs/10.1126/science.adh1174`.

[201] Thomas Monks, Alison Harper, Anastasia Anagnostou, and Simon JE Taylor. Open science for computer simulation. 2022.

[202] Miguel Angel Zamora Mora, Momchil Peychev, Sehoon Ha, Martin Vechev, and Stelian Coros. Pods: Policy optimization via differentiable simulation. In *International Conference on Machine Learning*, pages 7805–7817. PMLR, 2021.

[203] Eric Müller, Elias Arnold, Oliver Breitwieser, Milena Czierlinski, Arne Emmel, Jakob Kaiser, Christian Mauch, Sebastian Schmitt, Philipp Spilger, Raphael Stock, et al. A scalable approach to modeling on accelerated neuromorphic hardware. *Frontiers in neuroscience*, 16:884128, 2022.

[204] Thomas Müller, Brian Mcwilliams, Fabrice Rousselle, Markus Gross, and Jan Novák. Neural importance sampling. *ACM Trans. Graph.*, 38(5), oct 2019. ISSN 0730-0301. doi: 10.1145/3341156. URL `https://doi.org/10.1145/3341156`.

[205] Daniel Musekamp, Marimuthu Kalimuthu, David Holzmüller, Makoto Takamoto, and Mathias Niepert. Active learning for neural pde solvers, 2024. URL `https://arxiv.org/abs/2408.01536`.

[206] Mina Naghshnejad and Mukesh Singhal. A hybrid scheduling platform: a runtime prediction reliability aware scheduling platform to improve hpc scheduling performance. *The Journal of Supercomputing*, 76(1):122–149, Jan 2020. ISSN 1573-0484. doi: 10.1007/s11227-019-03004-3. URL `https://doi.org/10.1007/s11227-019-03004-3`.

[207] Sai Kiran Narayanasamy. Automating mechanism design with program synthesis. *Proc. Automated Learning Agents Workshop*, 2022. URL `https://par.nsf.gov/biblio/10391908`.

[208] Yurii Nesterov and Vladimir Spokoiny. Random gradient-free minimization of convex functions. *Foundations of Computational Mathematics*, 17(2):527–566, 2017.

[209] Ngan Nguyen, Feng Liang, Dominik Engel, Ciril Bohak, Peter Wonka, Timo Ropinski, and Ivan Viola. Differentiable electron microscopy simulation: Methods and applications for visualization. *arXiv preprint arXiv:2205.04464*, 2022.

[210] Quang Anh Pham Nguyen, Philipp Andelfinger, Wen Jun Tan, Wentong Cai, and Alois Knoll. Transitioning spiking neural network simulators to heterogeneous hardware. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 31(2):1–26, 2021.

[211] David M Nicol. The cost of conservative synchronization in parallel discrete event simulations. *Journal of the ACM*, 40(2):304–333, April 1993. ISSN 0004-5411. doi: 10.1145/151261.151266.

[212] Lars Niedermeier, Kexin Chen, Jinwei Xing, Anup Das, Jeffrey Kopsick, Eric Scott, Nate Sutton, Killian Weber, Nikil Dutt, and Jeffrey L Krichmar. Carlsim 6: an open source library for large-scale, biologically detailed spiking neural network simulation. In *2022 International Joint Conference on Neural Networks (IJCNN)*, pages 1–10. IEEE, 2022.

[213] Theo X Olausson, Alex Gu, Ben Lipkin, Cedegao E Zhang, Armando Solar-Lezama, Joshua B Tenenbaum, and Roger P Levy. Linc: A neurosymbolic approach for logical reasoning by combining language models with first-order logic provers. In *The 2023 Conference on Empirical Methods in Natural Language Processing*.

[214] Liangming Pan, Alon Albalak, Xinyi Wang, and William Yang Wang. Logic-lm: Empowering large language models with symbolic solvers for faithful logical reasoning. In *The 2023 Conference on Empirical Methods in Natural Language Processing*.

[215] George Papamakarios and Iain Murray. Fast ε-free inference of simulation models with bayesian conditional density estimation. *Advances in neural information processing systems*, 29, 2016.

[216] George Papamakarios, David Sterratt, and Iain Murray. Sequential neural likelihood: Fast likelihood-free inference with autoregressive flows. In *The 22nd international conference on artificial intelligence and statistics*, pages 837–848. PMLR, 2019.

[217] Kyriakos Paraskevas. Enabling direct-access global shared memory for distributed heterogeneous computing. 2023.

[218] Hyungwook Park and Paul A Fishwick. A gpu-based application framework supporting fast discrete-event simulation. *Simulation*, 86(10):613–628, 2010.

[219] Kanghee Park, Jiayu Wang, Taylor Berg-Kirkpatrick, Nadia Polikarpova, and Loris D'Antoni. Grammar-aligned decoding. *arXiv preprint arXiv:2405.21047*, 2024.

[220] Christian Pehle, Sebastian Billaudelle, Benjamin Cramer, Jakob Kaiser, Korbinian Schreiber, Yannik Stradmann, Johannes Weis, Aron Leibfried, Eric Müller, and Johannes Schemmel. The brainscales-2 accelerated neuromorphic system with hybrid plasticity. *Frontiers in Neuroscience*, 16:795876, 2022.

[221] Raul P Pelaez, Guillem Simeon, Raimondas Galvelis, Antonio Mirarchi, Peter Eastman, Stefan Doerr, Philipp Thölke, Thomas E Markland, and Gianni De Fabritiis. Torchmd-net 2.0: Fast neural network potentials for molecular simulations. *Journal of Chemical Theory and Computation*, 2024.

[222] Alessandro Pellegrini. Optimizing memory management for optimistic simulation with reinforcement learning. In *2016 International Conference on High Performance Computing & Simulation (HPCS)*, pages 26–33. IEEE, 2016.

[223] Alessandro Pellegrini and Francesco Quaglia. NUMA time warp. In *Proceedings of the 3rd ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*, SIGSIM PADS' 15, pages 59–70, New York, NY, USA, June 2015. ACM. ISBN 9781450335836. doi: 10.1145/2769458.2769479.

[224] Ayyappasamy Sudalaiyadum Perumal, Zihao Wang, Giulia Ippoliti, Falco CMJM van Delft, Lila Kari, and Dan V Nicolau. As good as it gets: a scaling comparison of dna computing, network biocomputing, and electronic computing approaches to an np-complete problem. *New Journal of Physics*, 23(12):125001, 2021.

[225] Kalyan S Perumalla. Discrete-event execution alternatives on general purpose graphical processing units (gpgpus). In *20th Workshop on Principles of Advanced and Distributed Simulation (PADS'06)*, pages 74–81. IEEE, 2006.

[226] Kalyan S Perumalla. Efficient execution on GPUs of field-based vehicular mobility models. In *Proceedings of the 22nd Workshop on Principles of Advanced and Distributed Simulation*, PADS'08, page 154, Piscataway, NJ, USA, June 2008. IEEE. ISBN 9780769531595. doi: 10.1109/pads.2008.36.

[227] Kalyan S Perumalla. *Introduction to reversible computing*. CRC Press, 2013.

[228] Brenden K Petersen, Mikel Landajuela, T Nathan Mundhenk, Claudio P Santiago, Soo K Kim, and Joanne T Kim. Deep symbolic regression: Recovering mathematical expressions from data via risk-seeking policy gradients. *arXiv preprint arXiv:1912.04871*, 2019.

[229] Fabio Petroni, Tim Rocktäschel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, Alexander H Miller, and Sebastian Riedel. Language models as knowledge bases? *arXiv preprint arXiv:1909.01066*, 2019.

[230] FW Pfeiffer. Automatic differentiation in prose. *ACM SIGNUM Newsletter*, 22(1):2–8, 1987.

[231] Andrea Piccione, Philipp Andelfinger, and Alessandro Pellegrini. Hybrid speculative synchronisation for parallel discrete event simulation. In *Proceedings of the 2023 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*, SIGSIM-PADS '23, pages 84–95, New York, NY, USA, June 2023. Association for Computing Machinery. ISBN 9798400700309. doi: 10.1145/3573900.3591124.

[232] Gianluigi Pillonetto, Aleksandr Aravkin, Daniel Gedon, Lennart Ljung, Antônio H Ribeiro, and Thomas B Schön. Deep networks for system identification: a survey. *Automatica*, 171:111907, 2025.

[233] Ernesto Posse, Jean-Sébastien Bolduc, and Hans Vangheluwe. Generation of devs modelling and simulation environments. In *Proceedings of the 2003 Summer Computer Simulation Conference*, pages 139–146, 2003.

[234] Dennis Prangle and Cecilia Viscardi. Distilling importance sampling for likelihood free inference. *Journal of Computational and Graphical Statistics*, 32(4):1461–1471, 2023. doi: 10.1080/10618600.2023.2175688. URL `https://doi.org/10.1080/10618600.2023.2175688`.

[235] Dennis Prangle and Cecilia Viscardi. Distilling importance sampling for likelihood free inference. *Journal of Computational and Graphical Statistics*, 32(4):1461–1471, 2023.

[236] Yewen Pu, Kevin Ellis, Marta Kryven, Josh Tenenbaum, and Armando Solar-Lezama. Program synthesis with pragmatic communication. *Advances in neural information processing systems*, 33:13249–13259, 2020.

[237] Yewen Pu, Saujas Vaduguru, Priyan Vaithilingam, Elena Glassman, and Daniel Fried. Amortizing pragmatic program synthesis with rankings. *arXiv preprint arXiv:2407.02499*, 2024.

[238] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.

[239] Herschel Rabitz and Ömer F. Aliş. General foundations of high-dimensional model representations. *Journal of Mathematical Chemistry*, 25(2):197–233, Jun 1999. ISSN 1572-8897. doi: 10.1023/A:1019188517934. URL `https://doi.org/10.1023/A:1019188517934`.

[240] Christopher Rackauckas, Yingbo Ma, Julius Martensen, Collin Warner, Kirill Zubov, Rohit Supekar, Dominic Skinner, Ali Ramadhan, and Alan Edelman. Universal differential equations for scientific machine learning. *arXiv preprint arXiv:2001.04385*, 2020.

[241] Bharath Raghavan, Mirko Paulikat, Katya Ahmad, Lara Callea, Andrea Rizzi, Emiliano Ippoliti, Davide Mandelli, Laura Bonati, Marco De Vivo, and Paolo Carloni. Drug design in the exascale era: a perspective from massively parallel qm/mm simulations. *Journal of chemical information and modeling*, 63(12):3647–3658, 2023.

[242] Hamed Rahimian and Sanjay Mehrotra. Distributionally robust optimization: A review. *arXiv preprint arXiv:1908.05659*, 2019.

[243] Shafiur Rahman, Nael Abu-Ghazaleh, and Walid Najjar. PDES-a: Accelerators for parallel discrete event simulation implemented on FPGAs. *ACM transactions on modeling and computer simulation: a publication of the Association for Computing Machinery*, 29 (2):1–25, April 2019. ISSN 1049-3301,1558-1195. doi: 10.1145/3302259.

[244] Steven F. Railsback and Volker Grimm. *Agent-Based and Individual-Based Modeling: A Practical Introduction*. Princeton University Press, 2011. ISBN 0691136742. URL `http://www.jstor.org/stable/j.ctt7sns7`.

[245] Milad Ramezankhani, Mehrtash Harandi, Rudolf Seethaler, and Abbas S Milani. Smart manufacturing under limited and heterogeneous data: a sim-to-real transfer learning with convolutional variational autoencoder in thermoforming. *International Journal of Computer Integrated Manufacturing*, 37(1-2):18–36, 2024.

[246] Saman Razavi, Bryan A Tolson, L Shawn Matott, Neil R Thomson, Angela MacLean, and Frank R Seglenieks. Reducing the computational cost of automatic calibration through model preemption. *Water Resources Research*, 46(11), 2010.

[247] Chao Ren, Younes Aoues, Didier Lemosse, and Eduardo Souza De Cursi. Ensemble of surrogates combining kriging and artificial neural networks for reliability analysis with local goodness measurement. *Structural Safety*, 96:102186, 2022.

[248] Paul Richmond, Robert Chisholm, Peter Heywood, Mozhgan Kabiri Chimeh, and Matthew Leach. Flame gpu 2: A framework for flexible and performant agent based simulation on gpus. *Software: Practice and Experience*, 53(8):1659–1680, 2023.

[249] Ryan Riegel, Alexander Gray, Francois Luus, Naweed Khan, Ndivhuwo Makondo, Ismail Yunus Akhalwaya, Haifeng Qian, Ronald Fagin, Francisco Barahona, Udit Sharma, et al. Logical neural networks. *arXiv preprint arXiv:2006.13155*, 2020.

[250] Javier Robledo-Moreno, Mario Motta, Holger Haas, Ali Javadi-Abhari, Petar Jurcevic, William Kirby, Simon Martiel, Kunal Sharma, Sandeep Sharma, Tomonori Shirakawa, Iskandar Sitdikov, Rong-Yang Sun, Kevin J. Sung, Maika Takita, Minh C. Tran, Seiji Yunoki, and Antonio Mezzacapo. Chemistry beyond exact solutions on a quantum-centric supercomputer, 2024.

[251] Alberto Rodrigues da Silva. Model-driven engineering: A survey supported by the unified conceptual model. *Computer languages, systems & structures*, 43:139–155, October 2015. ISSN 1477-8424,1873-6866. doi: 10.1016/j.cl.2015.06.001.

[252] Julia Rudnitckaia. Process mining. data science in action. *University of Technology, Faculty of Information Technology*, pages 1–11, 2016.

[253] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[254] Maximiliano A. Sacco, Juan J. Ruiz, Manuel Pulido, and Pierre Tandeo. Evaluation of machine learning techniques for forecast uncertainty quantification. *Quarterly Journal of the Royal Meteorological Society*, 148(749):3470–3490, oct 2022. ISSN 1477-870X. doi: 10.1002/qj.4362. URL `http://dx.doi.org/10.1002/qj.4362`.

[255] Subham Sahoo, Christoph Lampert, and Georg Martius. Learning equations for extrapolation and control. In *International Conference on Machine Learning*, pages 4442–4450. Pmlr, 2018.

[256] PM Scala, MA Mujica Mota, RS Felix Patron, and A Murrieta Mendoza. Airport passenger flow prediction using simulation data farming and machine learning. In *Proceedings of the 33rd European Modeling & Simulation Symposium (EMSS 2021)*, pages 165–172. CAL-TEK, 2021.

[257] Marius Schlegel and Kai-Uwe Sattler. Management of machine learning lifecycle artifacts: A survey. *ACM SIGMOD Record*, 51(4):18–35, 2023.

[258] Samuel Schoenholz and Ekin Dogus Cubuk. Jax md: a framework for differentiable physics. *Advances in Neural Information Processing Systems*, 33:11428–11441, 2020.

[259] Catherine D. Schuman, Thomas E. Potok, Robert M. Patton, J. Douglas Birdwell, Mark E. Dean, Garrett S. Rose, and James S. Plank. A survey of neuromorphic computing and neural networks in hardware, 2017. URL `https://arxiv.org/abs/1705.06963`.

[260] Lynne Serré and Maude Amyot-Bourgeois. An application of automated machine learning within a data farming process. In *2022 Winter Simulation Conference (WSC)*, pages 2013–2024. IEEE, 2022.

[261] Mojtaba Shahin, M Ali Babar, and Muhammad Aufeef Chauhan. Architectural design space for modelling and simulation as a service: a review. *Journal of Systems and Software*, 170:110752, 2020.

[262] N Shavit and I Lotan. Skiplist-based concurrent priority queues. In *Proceedings of the 14th International Parallel and Distributed Processing Symposium*, IPDPS, pages 263–268, Piscataway, NJ, USA, May 2000. IEEE. doi: 10.1109/IPDPS.2000.845994.

[263] Yiren Shen, Harsh C. Patel, Zan Xu, and Juan J. Alonso. *Application of Multi-Fidelity Transfer Learning with Autoencoders for Efficient Construction of Surrogate Models*. doi: 10.2514/6.2024-0013. URL `https://arc.aiaa.org/doi/abs/10.2514/6.2024-0013`.

[264] Parshin Shojaee, Kazem Meidani, Amir Barati Farimani, and Chandan Reddy. Transformer-based planning for symbolic regression. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 45907–45919. Curran Associates, Inc., 2023. URL `https://proceedings.neurips.cc/paper_files/paper/2023/file/8ffb4e3118280a66b192b6f06e0e2596-Paper-Conference.pdf`.

[265] Martin Sipka, Johannes C. B. Dietschreit, Lukáš Grajciar, and Rafael Gomez-Bombarelli. Differentiable simulations for enhanced sampling of rare events. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 31990–32007. PMLR, 23–29 Jul 2023. URL `https://proceedings.mlr.press/v202/sipka23a.html`.

[266] Daniel Dominic Sleator and Robert Endre Tarjan. Self-adjusting binary search trees. *Journal of the ACM*, 32(3):652–686, July 1985. ISSN 0004-5411. doi: 10.1145/3828.3835.

[267] Kaya Stechly, Matthew Marquez, and Subbarao Kambhampati. Gpt-4 doesn't know it's wrong: An analysis of iterative prompting for reasoning problems. *arXiv preprint arXiv:2310.12397*, 2023.

[268] Jeffrey S Steinman. SPEEDES: Synchronous parallel environment for emulation and discrete event simulation. In Vijay K Madisetti, David Nicol, and Richard M Fujimoto, editors, *Advances in Parallel and Distributed Simulation*, PADS '91, pages 1111–1115, San Diego, CA, USA, 1991. Society for Computer Simulation.

[269] Keiran Suchak, Minh Kieu, Yannick Oswald, Jonathan A Ward, and Nick Malleson. Coupling an agent-based model and an ensemble kalman filter for real-time crowd modelling. *Royal Society Open Science*, 11(4):231553, 2024.

[270] Hyung Ju Suh, Max Simchowitz, Kaiqing Zhang, and Russ Tedrake. Do differentiable simulators give better policy gradients? In *International Conference on Machine Learning*, pages 20668–20696. PMLR, 2022.

[271] Jennifer J. Sun, Megan Tjandrasuwita, Atharva Sehgal, Armando Solar-Lezama, Swarat Chaudhuri, Yisong Yue, and Omar Costilla-Reyes. Neurosymbolic programming for science, 2022. URL `https://arxiv.org/abs/2210.05050`.

[272] Herb Sutter et al. The free lunch is over: A fundamental turn toward concurrency in software. *Dr. Dobb's journal*, 30(3):202–210, 2005.

[273] Wen Jun Tan, Moon Gi Seok, and Wentong Cai. Automatic model generation and data assimilation framework for cyber-physical production systems. In *Proceedings of the 2023 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*, pages 73–83, 2023.

[274] Daniel Tang and Nick Malleson. Data assimilation with agent-based models using markov chain sampling. *Open Research Europe*, 2:70, 2022.

[275] Wai Teng Tang, Rick Siow Mong Goh, and Ian Li-Jin Thng. Ladder queue: An O(1) priority queue structure for large-scale discrete event simulation. *ACM Transactions on Modeling and Computer Simulation*, 15(3):175–204, July 2005. ISSN 1049-3301. doi: 10.1145/1103323.1103324.

[276] Simon JE Taylor, Tamas Kiss, Anastasia Anagnostou, Gabor Terstyanszky, Peter Kacsuk, Joris Costes, and Nicola Fantini. The cloudsme simulation platform and its applications: A generic multi-cloud platform for developing and executing commercial cloud-based simulations. *Future Generation Computer Systems*, 88:524–539, 2018.

[277] Wassim Tenachi, Rodrigo Ibata, and Foivos I. Diakogiannis. Deep symbolic regression for physics guided by units constraints: Toward the automated discovery of physical laws. *The Astrophysical Journal*, 959(2):99, dec 2023. doi: 10.3847/1538-4357/ad014c. URL `https://dx.doi.org/10.3847/1538-4357/ad014c`.

[278] Patricia Ternes, Jonathan A Ward, Alison Heppenstall, Vijay Kumar, Le-Minh Kieu, and Nick Malleson. Data assimilation and agent-based modelling: towards the incorporation of categorical agent parameters. *Open Research Europe*, 1, 2021.

[279] Timm Teubner, Christoph M Flath, Christof Weinhardt, Wil van der Aalst, and Oliver Hinz. Welcome to the era of chatgpt et al. the prospects of large language models. *Business & Information Systems Engineering*, 65(2):95–101, 2023.

[280] Andreas Tolk. Learning something right from models that are wrong: Epistemology of simulation. In *Concepts and methodologies for modeling and simulation: A tribute to Tuncer Ören*, pages 87–106. Springer, 2015.

[281] Rohit K. Tripathy and Ilias Bilionis. Deep uq: Learning deep neural network surrogate models for high dimensional uncertainty quantification. *Journal of Computational Physics*, 375:565–588, 2018. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2018.08.036. URL `https://www.sciencedirect.com/science/article/pii/S0021999118305655`.

[282] Spyridon Tsattalios, Ioannis Tsoukalas, Panagiotis Dimas, Panagiotis Kossieris, Andreas Efstratiadis, and Christos Makropoulos. Advancing surrogate-based optimization of time-expensive environmental problems through adaptive multi-model search. *Environmental Modelling & Software*, 162:105639, 2023.

[283] Adelinde M Uhrmacher, Peter Frazier, Reiner Hähnle, Franziska Klügl, Fabian Lorig, Bertram Ludäscher, Laura Nenzi, Cristina Ruiz-Martin, Bernhard Rumpe, Claudia Szabo, et al. Context, composition, automation, and communication: The c2ac roadmap for modeling and simulation. *ACM Transactions on Modeling and Computer Simulation*, 34(4):1–51, 2024.

[284] Michael D. Vander Wal, Ryan G. McClarren, and Kelli D. Humbird. Transfer learning of high-fidelity opacity spectra in autoencoders and surrogate models. *IEEE Transactions on Plasma Science*, 51(1):109–119, 2023. doi: 10.1109/TPS.2022.3227506.

[285] Eric Walter. *Identifiability of parametric models*. Elsevier, 2014.

[286] Boshi Wang, Xiang Yue, and Huan Sun. Can chatgpt defend its belief in truth? evaluating llm reasoning via debate. *arXiv preprint arXiv:2305.13160*, 2023.

[287] Congye Wang, Wilson Chen, Heishiro Kanagawa, Chris Oates, et al. Reinforcement learning for adaptive mcmc. *arXiv preprint arXiv:2405.13574*, 2024.

[288] Jun Wang and Carl Tropper. Optimizing time warp simulation with reinforcement learning techniques. In *2007 Winter Simulation Conference*, pages 577–584. IEEE, 2007.

[289] Minghao Wang and Xiaolin Hu. Data assimilation in agent based simulation of smart environments using particle filters. *Simulation Modelling Practice and Theory*, 56:36–54, 2015.

[290] Qiqi Wang, Hongjie Zhang, Cheng Qu, Yu Shen, Xiaohui Liu, and Jing Li. Rlschert: An hpc job scheduler using deep reinforcement learning and remaining time prediction. *Applied Sciences*, 11(20), 2021. ISSN 2076-3417. doi: 10.3390/app11209448. URL `https://www.mdpi.com/2076-3417/11/20/9448`.

[291] Zheng Wang, Yili Xia, Shanxiang Lyu, and Cong Ling. Reinforcement learning-aided markov chain monte carlo for lattice gaussian sampling. In *2021 IEEE Information Theory Workshop (ITW)*, pages 1–5. IEEE, 2021.

[292] Antoine Wehenkel, Jens Behrmann, Andrew C Miller, Guillermo Sapiro, Ozan Sener, Marco Cuturi, and Jörn-Henrik Jacobsen. Simulation-based inference for cardiovascular models. *arXiv preprint arXiv:2307.13918*, 2023.

[293] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

[294] Martin H Weik. The eniac story. *Ordnance*, 45(244):571–575, 1961.

[295] Pia Wilsdorf, Jakob Heller, Kai Budde, Julius Zimmermann, Tom Warnke, Christian Haubelt, Dirk Timmermann, Ursula van Rienen, and Adelinde M Uhrmacher. A model-driven approach for conducting simulation experiments. *Applied Sciences*, 12(16):7977, 2022.

[296] Pia Wilsdorf, Anja Wolpers, Jason Hilton, Fiete Haack, and Adelinde Uhrmacher. Automatic reuse, adaption, and execution of simulation experiments via provenance patterns. *ACM Transactions on Modeling and Computer Simulation*, 33(1-2):1–27, 2023.

[297] Di Wu, Helin Zhu, and Enlu Zhou. A bayesian risk approach to data-driven stochastic optimization: Formulations and asymptotics. *SIAM Journal on Optimization*, 28(2): 1588–1612, 2018.

[298] Jiajian Xiao, Philipp Andelfinger, David Eckhoff, Wentong Cai, and Alois Knoll. Exploring execution schemes for agent-based traffic simulation on heterogeneous hardware. In *Proceedings of the 22nd International Symposium on Distributed Simulation and Real Time Applications*, DS-RT '18, pages 1–10, Piscataway, NJ, USA, October 2018. IEEE Computer Society. ISBN 9781538650486. doi: 10.1109/DISTRA.2018.8601016.

[299] Jiajian Xiao, Philipp Andelfinger, Wentong Cai, Paul Richmond, Alois Knoll, and David Eckhoff. Openablext: An automatic code generation framework for agent-based simulations on cpu-gpu-fpga heterogeneous platforms. *Concurrency and Computation: Practice and Experience*, 32(21):e5807, 2020.

[300] Jiajian Xiao, Görkem Kilinç, Philipp Andelfinger, David Eckhoff, Wentong Cai, and Alois Knoll. Pedal to the bare metal: Road traffic simulation on FPGAs using high-level synthesis. In *Proceedings of the 2020 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*, SIGSIM-PADS'20, pages 117–121, New York, NY, USA, June 2020. ACM. ISBN 9781450375924. doi: 10.1145/3384441.3395979.

[301] Xu Xie, Alexander Verbraeck, and Feng Gu. Data assimilation in discrete event simulations-a rollback based sequential monte carlo approach. In *2016 Symposium on Theory of Modeling and Simulation (TMS-DEVS)*, pages 1–8. IEEE, 2016.

[302] Xiewei Xiong, Tong Zhu, Yun Zhu, Mengyao Cao, Jin Xiao, Li Li, Fei Wang, Chunhai Fan, and Hao Pei. Molecular convolutional neural networks with dna regulatory circuits. *Nature Machine Intelligence*, 4(7):625–635, 2022.

[303] Jundong Xu, Hao Fei, Liangming Pan, Qian Liu, Mong-Li Lee, and Wynne Hsu. Faithful logical reasoning via symbolic chain-of-thought. *arXiv preprint arXiv:2405.18357*, 2024.

[304] Chenxi Yang and Swarat Chaudhuri. Safe neurosymbolic learning with differentiable symbolic execution, 2022. URL `https://arxiv.org/abs/2203.07671`.

[305] Mengyue Yang, Furui Liu, Zhitang Chen, Xinwei Shen, Jianye Hao, and Jun Wang. Causalvae: Disentangled representation learning via neural structural causal models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9593–9602, 2021.

[306] Shuo Yang, Bas WA Bögels, Fei Wang, Can Xu, Hongjing Dou, Stephen Mann, Chunhai Fan, and Tom FA de Greef. Dna as a universal chemical substrate for computing and data storage. *Nature Reviews Chemistry*, 8(3):179–194, 2024.

[307] Feng Yao, Yiping Yao, Huangke Chen, Tianlin Li, Menglong Lin, and Xiaoxiong Zhang. An efficient virtual machine allocation algorithm for parallel and distributed simulation applications. *Concurrency and Computation: Practice and Experience*, 31(17):e5237, 2019.

[308] Esin Yavuz, James Turner, and Thomas Nowotny. Genn: a code generation framework for accelerated brain simulations. *Scientific reports*, 6(1):18854, 2016.

[309] Zhisheng Ye, Wei Gao, Qinghao Hu, Peng Sun, Xiaolin Wang, Yingwei Luo, Tianwei Zhang, and Yonggang Wen. Deep learning workload scheduling in gpu datacenters: A survey. *ACM Computing Surveys*, 56(6):1–38, 2024.

[310] Serhat Yeşilyurt and Anthony T. patera. Surrogates for numerical simulations; optimization of eddy-promoter heat exchangers. *Computer Methods in Applied Mechanics and Engineering*, 121(1):231–257, 1995. ISSN 0045-7825. doi: https://doi.org/10.1016/0045-7825(94)00684-F. URL https://www.sciencedirect.com/science/article/pii/004578259400684F.

[311] Xinhao Yi, Siwei Liu, Yu Wu, Douglas McCloskey, and Zaiqiao Meng. BPP: a platform for automatic biochemical pathway prediction. *Briefings in Bioinformatics*, 25(5):bbae355, 07 2024. ISSN 1477-4054. doi: 10.1093/bib/bbae355. URL https://doi.org/10.1093/bib/bbae355.

[312] Suleyman Yildirim, Alper Ekrem Murat, Murat Yildirim, and Suzan Arslanturk. Process knowledge driven change point detection for automated calibration of discrete event simulation models using machine learning. *arXiv preprint arXiv:2005.05385*, 2020.

[313] Srikanth B Yoginath and Kalyan S Perumalla. Efficient parallel discrete event simulation on cloud/virtual machine platforms. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 26(1):1–26, 2015.

[314] Daniel Zehe, Alois Knoll, Wentong Cai, and Heiko Aydt. Semsim cloud service: Large-scale urban systems simulation in the cloud. *Simulation Modelling Practice and Theory*, 58:157–171, 2015.

[315] Jinzhe Zeng, Duo Zhang, Denghui Lu, Pinghui Mo, Zeyu Li, Yixiao Chen, Marián Rynik, Li'ang Huang, Ziyao Li, Shaochen Shi, et al. Deepmd-kit v2: A software package for deep potential models. *The Journal of Chemical Physics*, 159(5), 2023.

[316] Di Zhang, Dong Dai, Youbiao He, Forrest Sheng Bao, and Bing Xie. Rlscheduler: An automated hpc batch job scheduler using reinforcement learning. In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–15, 2020. doi: 10.1109/SC41405.2020.00035.

[317] Dichang Zhang, Zexia Zhang, Christian Santoni, Ali Khosronejad, and Dimitris Samaras. Transfer learning in multi-fidelity surrogate modeling: A wind farm case. In *ICML 2024 AI for Science Workshop*, 2024. URL https://openreview.net/forum?id=yBTDCqNcan.

[318] Jinding Zhang, Jinzheng Kang, Kai Zhang, Liming Zhang, Piyang Liu, Xingyu Liu, Weijia Sun, and Guangyao Wang. An efficient transformer-based surrogate model with end-to-end training strategies for automatic history matching. *Geoenergy Science and Engineering*, 240:212994, 2024. ISSN 2949-8910. doi: https://doi.org/10.1016/j.geoen.2024.212994. URL https://www.sciencedirect.com/science/article/pii/S2949891024003646.

[319] Jun Zhang, Yi Isaac Yang, and Frank Noé. Targeted adversarial learning optimized sampling. *The journal of physical chemistry letters*, 10(19):5791–5797, 2019.

[320] Jun Zhang, Yao-Kun Lei, Zhen Zhang, Xu Han, Maodong Li, Lijiang Yang, Yi Isaac Yang, and Yi Qin Gao. Deep reinforcement learning of transition states. *Physical Chemistry Chemical Physics*, 23(11):6888–6895, 2021.

[321] Junzi Zhang, Jongho Kim, Brendan O'Donoghue, and Stephen Boyd. Sample efficient reinforcement learning with reinforce. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 10887–10895, 2021.

[322] Linfeng Zhang, De-Ye Lin, Han Wang, Roberto Car, and Weinan E. Active learning of uniformly accurate interatomic potentials for materials simulation. *Physical Review Materials*, 3(2):023804, 2019.

[323] Yuchang Zhang, Ruibin Bai, Rong Qu, Chaofan Tu, and Jiahuan Jin. A deep reinforcement learning based hyper-heuristic for combinatorial optimisation with uncertainties. *European Journal of Operational Research*, 300(2):418–427, 2022.

[324] Zirui Zhao, Wee Sun Lee, and David Hsu. Large language models as commonsense knowledge for large-scale task planning. *Advances in Neural Information Processing Systems*, 36, 2024.

[325] Jinghui Zhong, Wentong Cai, Michael Lees, and Linbo Luo. Automatic model construction for the behavior of human crowds. *Applied Soft Computing*, 56:368–378, 2017.

[326] Jinghui Zhong, Dongrui Li, Zhixing Huang, Chengyu Lu, and Wentong Cai. Data-driven crowd modeling techniques: A survey. *ACM Trans. Model. Comput. Simul.*, 32(1), January 2022. ISSN 1049-3301. doi: 10.1145/3481299. URL `https://doi.org/10.1145/3481299`.

[327] Changcong Zhou, Hanlin Zhang, Qi Chang, Xiaokang Song, and Chen Li. An adaptive ensemble of surrogate models based on hybrid measure for reliability analysis. *Structural and Multidisciplinary Optimization*, 65(1):16, 2022.

[328] Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. Large language models are human-level prompt engineers. *arXiv preprint arXiv:2211.01910*, 2022.

[329] Shengyu Zhu, Ignavier Ng, and Zhitang Chen. Causal discovery with reinforcement learning. *arXiv preprint arXiv:1906.04477*, 2019.