Towards Quantum Software for Quantum Simulation

Maja Franz, Lukas Schmidbauer
{maja.franz,lukas.schmidbauer}@othr.de
Technical University of Applied Sciences Regensburg
Regensburg, Germany

Joshua Ammermann, Ina Schaefer
{joshua.ammermann,ina.schaefer}@kit.edu
Karlsruhe Institute of Technology
Karlsruhe, Germany

Wolfgang Mauerer
wolfgang.mauerer@othr.de
Technical University of Applied Sciences Regensburg/Siemens AG, Foundational Technologies
Regensburg/Munich, Germany

Abstract

Quantum simulation is a leading candidate for demonstrating practical quantum advantage over classical computation, as it is believed to provide exponentially more compute power than any classical system. It offers new means of studying the behaviour of complex physical systems, for which conventionally software-intensive simulation codes based on numerical high-performance computing are used. Instead, quantum simulations map properties and characteristics of subject systems, for instance chemical molecules, onto quantum devices that then mimic the system under study.

Currently, the use of these techniques is largely limited to fundamental science, as the overall approach remains tailored for specific problems: We lack infrastructure and modelling abstractions that are provided by the software engineering (SE) community for other computational domains.

In this paper, we identify critical gaps in the quantum simulation software stack – particularly the absence of general-purpose frameworks for model specification, Hamiltonian construction, and hardware-aware mappings. We advocate for a modular model-driven engineering (MDE) approach that supports different types of quantum simulation (digital and analogue), and facilitates automation, performance evaluation, and reusability. Through an example from high-energy physics, we outline a vision for a quantum simulation framework capable of supporting scalable, cross-platform simulation workflows.

Keywords

Quantum Simulation, Quantum Software, MDE

1 Introduction

Quantum simulation is widely regarded as one of the most promising near-term applications of quantum computing [3, 20, 1, 12]. Conceptionally, quantum simulation is similar to a small-scale model of an aircraft in a wind tunnel: instead of utilising intricate algorithms and numerically solving the fluid-dynamical equations of motion, the model itself experiences the phenomena under study [14]. Similarly, utilising the dynamical capabilities of a programmable quantum system (in other words, a quantum computer) to learn about the dynamics of an appropriately mapped subject system, offers a path to tackle problems that are classically intractable [19, 29].

However, the *software infrastructure* and modelling frameworks required to systematically develop, deploy, and analyse these simulations remain significantly underdeveloped, as we discuss in this work. The state-of-the-art of quantum simulation is largely model-based: A subject Hamiltonian, which describes the dynamics of a physical target system gets mapped onto quantum computers by manually constructing models to approximate the target dynamics (see, *e.g.*, Refs. [30, 33]). This approach is not only central to physics-oriented applications but also underpins the potential for exponential quantum advantage in a broad class of algorithmic primitives based on quantum singular value transformation (QSVT) [10]. However, in contrast to quantum optimisation or quantum machine learning, where well-established algebraic modelling techniques and domain-specific modelling languages (DSMLs) enable **automated transformation** into quantum-executable formats [18, 28, 26, 25], the process of preparing a quantum simulation remains largely manual, problem-specific, and hardware-dependent. The absence of a generic software stack severely limits the ability to explore the space of quantum simulation strategies, which makes it difficult to **generalise implementations**, quantify trade-offs between alternative mappings, or benchmark against classical methods. Additionally, the challenges are exacerbated by intricacies of quantum hardware that require substantial customisation [3].

Furthermore, when considering classical simulations of physical systems, empirical benchmarking and performance analysis are routine practices [8]. However, for quantum systems such assessments are hindered by the lack of robust **simulation and benchmarking tooling**. Existing classical methods such as Monte Carlo simulations face challenges like the sign problem [29], yet they benefit from decades of software development. In contrast, quantum simulations – though theoretically immune to certain classical limitations – often lack the practical scaffolding to exploit this advantage meaningfully.

To move beyond isolated demonstrations, the field must adopt a more **systematic and modular** approach to simulation design. Specifically, it is necessary to: (1) formalise the construction of effective model Hamiltonians from physical theories [19], (2) automate the mapping of these models to quantum hardware representations, and (3) develop standardised simulation workflows that allow for scalable execution, validation, and performance estimation. Building on the general quantum SE roadmap presented in Ref. [21], this work outlines the initial stages of developing a quantum simulation software framework, an area that has received limited attention in previous studies. We identify key stages in a simulation pipeline, from model specification and Hamiltonian design to hardware compilation, and explore open challenges in the pathway to systematic software support. Ultimately, enabling a software stack for quantum simulation – including domain-specific abstractions, modelling languages, and tooling for mapping and optimisation – is critical for unlocking the full potential of this physics-driven quantum approach. In the following, we outline the fundamentals of quantum simulation in Sec. 2. Sec. 3 then describes our vision of a general quantum simulation framework at an example from the high-energy

physics domain, which aims at a MDE [11] approach to quantum simulations. Sec. 4 lists open challenges.

2 Quantum Simulation

The (classically hard to solve) time evolution of (quantum) systems (*e.g.*, wind tunnel, electron interactions) is governed by a Hamiltonian $\hat{H}_{\text{sys}}$ that contains all information about the physics of that system. Starting at the systems initial state, configurable quantum simulators (*i.e.*, quantum "computers") can mimic this time evolution [16, 31], expressed by the unitary operator $\hat{U}_{\text{sys}}(t) = \exp\left(-i\int_0^t \hat{H}_{\text{sys}}(t)\mathrm{d}t\right)$. Importantly, the equation connects unitary gates $\hat{U}_{\text{sys}}$, the usual objects of study in quantum computing, with Hamiltonians $\hat{H}_{\text{sys}}$ (see left part of Fig. 1). We discuss two principal approaches (*i.e.*, digital and analogue) that realise this time evolution in the following (see Daley et al. [8] for a detailed discussion).

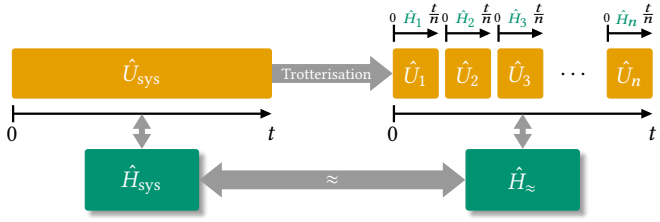

Figure 1: Continuous vs. discrete (Trotterised) time evolution. Left: The continuous dynamic of a physical systems is described by the Hamiltonian $\hat{H}_{\text{sys}}$ starting at an initial state at time 0, which then evolves under the corresponding unitary $\hat{U}_{\text{sys}}$ to a target state at time t. Right: Continuous time evolution is discretised (*Trotterised*) into local one- and two-qubit gates that can be executed on universal, gate-based quantum hardware, yet in the general case results in a time evolution under an approximated Hamiltonian $\hat{H}_{\approx}$.

2.1 Digital Quantum Simulation

For many physical systems, $\hat{H}_{\text{sys}}$ can be expressed as a sum of (non-commuting) k_j-local interaction terms: $\hat{H}_{\text{sys}} = \sum_j \hat{H}_j$. To address the non-commutativity in $\hat{H}_j$, a *Trotterisation* applies the k_j-local unitaries $\hat{U}_j = e^{-i\hat{H}_j(t/n)}$ n times over time slices t/n:

$$\hat{U}_{\text{sys}}(t) = e^{-i\sum_j \hat{H}_j(t)} = \underbrace{\left(\prod_j e^{-i\hat{H}_j(t/n)}\right)^n}_{\hat{U}_{\approx}(t)=\exp(-i\int_0^t \hat{H}_{\approx}(t)\mathrm{d}t)} + \underbrace{O\left(\frac{t^2}{n}\right)}_{\text{Approximation error}}, \quad (1)$$

where $\hat{U}_{\approx}$ approximates time evolution under an implicit Hamiltonian $\hat{H}_{\approx}$ (right part of Fig. 1). This decouples executing hardware from simulated dynamics for error-corrected systems, and can implement a large class of Hamiltonians, although discretisation leads to approximation errors and poorer scalability. Digital quantum simulation is universal as it allows for approximating arbitrary single- and two-qubit gates to arbitrary accuracy. This enables simulating the dynamics of quantum systems that may be fundamentally different from the simulating hardware. The resulting unitary gate instructions can be formulated using existing DSMLs [7, 23, 6] that can be executed on gate-based quantum computers. As is common in the circuit model, a large number of high-fidelity qubits and deep circuits are typically required. Error mitigation techniques allow for limited execution on noisy intermediate-scale quantum (NISQ) machines, but not yet at scale and precision needed for practical relevance [16].

2.2 Analogue Quantum Simulation

In contrast to digital simulation, analogue quantum simulation [14] aims to closely mimic the characteristics of the simulated quantum model by evolving under a simulator Hamiltonian $\hat{H}_{\text{sim}}$ that is similar to the simulated system Hamiltonian $\hat{H}_{\text{sys}}$. By directly mapping $\hat{H}_{\text{sys}} \mapsto \hat{H}_{\text{sim}}$ the time evolution proceeds under the natural Hamiltonian evolution of the simulator Hamiltonian and corresponding hardware. In particular in the context of physics-related questions, such as applications from high-energy physics and cosmology, analogue quantum simulation seen as proof of (often exponential) quantum advantages over the best known classical techniques [4, 8, 20]. The direct implementation of $\hat{H}_{\text{sim}}$ on a quantum system is promising in terms of scalability, but also entails limitations due to physical restrictions and non-universality of operations [31] that restrict flexibility. When $\hat{H}_{\text{sys}}$ is structurally similar to $\hat{H}_{\text{sim}}$, efficient mappings can be determined comparatively easily, while it remains a major challenge for more general problems. Although efforts in digital simulation have started to explore general algorithmic approaches [27], analogue simulations still rely heavily on problem-specific mappings that are hard to scale or reuse. Analogue simulation lacks modelling tools and formal abstractions that have accelerated progress in other areas, typically based on ideas from software engineering.

Lamata *et al.* [16] suggest a mixed approach where analogue blocks provide scalability by reducing gate-counts and hence sources of noise, while digital steps amplify the variety of possible operations. This may be useful on the way towards error-correcting codes to achieve usable results before perfect hardware is available.

3 Vision: Quantum Simulation Framework

Current quantum simulations often require manual processes for translating models into executable programs on quantum hardware. Instead, we propose a MDE [11] approach to quantum simulation, where high-level, abstract models guide automated transformations from theory to implementation. Using a concrete example, in which a theory from high-energy physics is simulated in an analogue mode, we highlight both the demands on quantum software and the deficiencies in today's tooling. As an instance of the software stack presented in Ref. [2], Fig. 2 illustrates our envisioned quantum simulation stack, tracing the transformation from the physical model to its realisation on configurable quantum hardware.

3.1 Example: Simulating a Lattice Gauge Theory

In our example we demonstrate the software requirements for simulating a U(1) Lattice Gauge Theory (LGT) [27], using an array of neutral atoms in an optical lattice as a simulation platform. This follows the simulation methodologies developed in [30, 33].

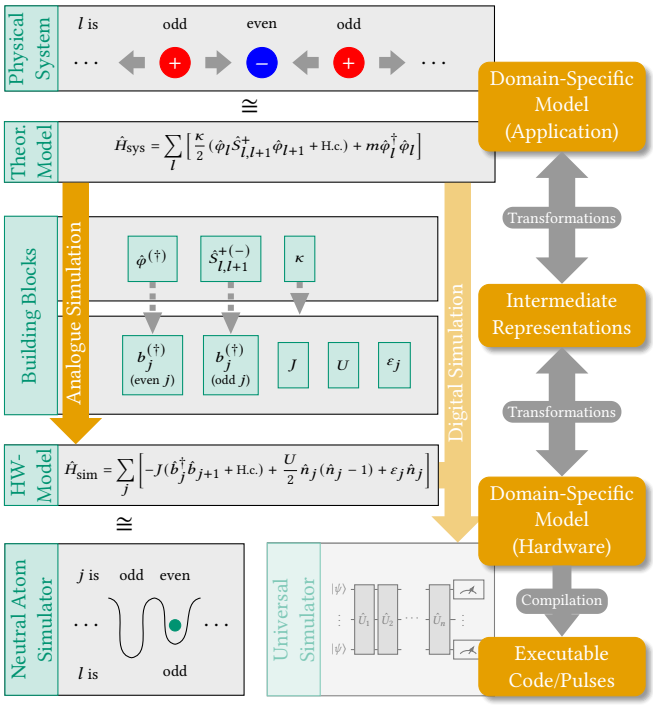

Figure 2: Envisioned quantum simulation framework illustrated for one-dimensional quantum electrodynamics (left), and suggested software abstraction layers (right).

A mathematical description of the target physical system, the **theoretical model**, captures the physical processes governing the system. In our case, a one-dimensional U(1) gauge theory serves as effective toy model for quantum electrodynamics (QED). While we cannot provide comprehensive physical details (see, *e.g.*, Ref. [27]), suffice it to say the Hamiltonian $\hat{H}_{\text{sys}}$ (*cf.* Fig. 2) comprises (a) matter field operators $\hat{\varphi}_l^{(\dagger)}$, where $\hat{\varphi}_l^\dagger$ creates a Fermion and $\hat{\varphi}_l$ annihilates one at lattice site l with mass m, and (b) gauge fields described by spin-$\frac{1}{2}$ operators $\hat{S}^{+(-)}_{l,l+1}$ on the links between neighbouring sites. Local Gauss law constraints, enforced via coupling κ, ensure physical constraints.

All ingredients are tied to the underlying physics, and share little commonality with computer science thinking.

For execution, Hamiltonian $\hat{H}_{\text{sys}}$ is mapped to Hamiltonian $\hat{H}_{\text{sim}}$ (*cf.* Fig. 2) governed by a **hardware model** (*i.e.*, in this case the *Bose-Hubbard* model), which can experimentally be realised with an array of neutral atoms. Term J in $\hat{H}_{\text{sim}}$ describes the "hopping" or tunnelling of atoms between adjacent lattice sites: $\hat{b}_{j+1}$ removes an atom at site $j+1$, and $\hat{b}_j^\dagger$ creates one at site j. The term leading with U represents the strength of the on-site interaction with $\hat{n}_j = \hat{b}_j^\dagger \hat{b}_j$, which is relevant when more than one atom is located on a single site. The ε_j term represents a energy offset to represent lattice potentials and suppress long-range tunnelling along the 1D chain.

In a **neutral atom simulator** model parameters can be tuned through laser intensity or magnetic/optical fields [9], such that the simulator Hamiltonian $\hat{H}_{\text{sim}}$ closely approximates $\hat{H}_{\text{sys}}$, preserving physical local constraints [30, 33]. A parity encoding can map the target model onto the simulator (*cf.* lower left part of Fig. 2): (1) even-indexed sites represent Fermionic matter fields $\hat{\varphi}_l$ and (2) odd ones gauge field links via spin operators. Hamiltonian $\hat{H}_{\text{sim}}$ is the lowest level of abstraction from a user's point of view, but control parameters and pulses must be derived for execution. While digital gates are firmly established in frameworks [23, 6, 7], pulse-level abstractions [22, 7, 17] remain evanescent, depend on hardware-specifics, and are not widely supported by hardware vendors. Work on analogue quantum simulation remains largely experimental, and lacks intermediate representations of Hamiltonians that can be transformed into pulse-level representations [24]. This suggests direct access to executing systems is required in the sense of *individual experiments*, rather than quantum *computers*.

Alternative Simulation Realisations. Analogue simulation is tightly coupled to the experimental platform, and requires bespoke configuration of physical parameters without enjoying support from software abstractions. Multiple mappings of the Bose-Hubbard model to other platform have been proposed, for instance trapped ions [15] or digital simulation via Trotterisation [27] that approximates continuous dynamics using gate-based circuits. This highlights diversity of platforms and methods, and the absence of a unified software framework capable of targeting multiple backends from a single high-level model. A corresponding interface and the integration with programmable quantum simulators are open challenges.

3.2 Need for Automation and Abstraction

Translating theoretical building blocks into a quantum simulation pipeline presents significant complexity and calls for automated, reusable, and programmable solutions. This motivates the use of an MDE approach and software abstractions that go beyond extensions of traditional approaches like UML [32]. While the roadmap by Murillo et al. [21] recognises the need for MDE in quantum SE, it does not discuss quantum simulation. Following Carbonelli et al. [5], we argue these challenges are particularly relevant in the simulation domain, and see need to address the following:

Domain-specific abstractions of physical theories. To effectively model quantum systems, it is essential to capture the underlying physical theories—such as Hamiltonian dynamics or many-body interactions—through high-level, domain-specific abstractions. These abstractions serve as a bridge between theoretical formulations and their executable representations, enabling automation and reuse.

Scalable, hardware-independent intermediate representations. A key requirement for portability and scalability in quantum simulation is the use of intermediate representations that decouple the simulation logic from specific hardware architectures. These representations must be expressive enough to encode complex quantum operations while remaining amenable to optimization and analysis.

Code generation and hardware-specific mappings. Automated code generation tools must translate abstract models into efficient, hardware-compatible code, accounting for unique constraints and capabilities of quantum backends. This involves optimising for gate sets, qubit connectivity, and target hardware characteristics.

Methods for transforming between abstractions. Consistency across abstraction levels requires formal, well-defined transformations to ensure changes at one level propagate correctly to others, and

preserve semantic integrity. We advocate for DSMLs as model representations. Benefits extend beyond physics-centric domains, as quantum simulation plays a role in other domains (*e.g.*, wind tunnel testing or computational fluid dynamics [13]).

4 Open Challenges

Quantum simulation workflows are often ad hoc, and tailored to specific platforms. A significant gap remains in the systematic translation of high-level theoretical models into executable intermediate representations compatible with constraints and features of quantum hardware. This hinders general-purpose solutions, and raises research questions: **Which intermediate representations are required to capture the properties of a physical system, yet allow for automatic transformations, optimisations, and scheduling to target specific hardware constraints?** and **How can noise models, error mitigation, and correction strategies be formally integrated into quantum software toolchains, such that their influence on performance can be systematically measured?** Formal abstractions, intermediate representations for digital *and* analogue operations, and tools for automated mapping and optimisation across hardware platforms are required. Furthermore, quantum simulation spans a spectrum from fully digital (gate-based) to fully analogue approaches. No systematic framework to evaluate or compare different simulation strategies along this continuum is available. This lack of software support limits the ability to make informed choices about how to implement a given simulation task. We therefore ask: **How can software architecture and patterns help select among digital, analogue, and hybrid simulation strategies based on the characteristics of the model and available hardware?** and **What benchmarking tools and performance models are needed to evaluate quantum simulation implementations–including trade-offs between quantum and classical backends, and across the digital-analogue spectrum?** Addressing these challenges requires not only new benchmarking methodologies but also software that can model, simulate, and optimise across the full design space.

Acknowledgments

This work is supported by the German Federal Ministry of Research, Technology and Space (BMFTR), funding program *quantum technologies—from basic research to market*, grant 13N17387, by the High-Tech Agenda of the Free State of Bavaria, and by the German Research Foundation, grants MA9739/1-1 and SCHA1635/18-1.

References

[1] E. Altman et al. 2021. *Quantum simulators: architectures and opportunities*. *PRX Quantum*, 017003.
[2] J. Ammermann, W. Mauerer, and I. Schaefer. 2024. *Towards view-based development of quantum software*. *INFORMATIK 2024*. Gesellschaft für Informatik e.V., Bonn, 551–554.
[3] C. W. Bauer et al. 2023. *Quantum simulation for high-energy physics*. *PRX Quantum*, 4, 027001, 2.
[4] J. Bermejo-Vega et al. 2018. *Architectures for quantum simulation showing a quantum speedup*. *PRX*, 8, 2.
[5] I. Exman et al., (Eds.) 2024. *Challenges for quantum software engineering: an industrial application scenario perspective*. *Quantum Software: Aspects of Theory and System Design*. Springer Nature Switzerland, Cham, 311–335.
[6] Cirq developers. 2025. *Cirq*. (2025).
[7] A. Cross et al. 2022. *Openqasm 3: a broader and deeper quantum assembly language*. *ACM TQC*, 3, 3, Article 12, 50 pages.
[8] A. J. Daley et al. 2022. *Practical quantum advantage in quantum simulation*. *Nature*, 607, 7920, 667–676.
[9] I. H. Deutsch, G. K. Brennen, and P. S. Jessen. 2000. *Quantum computing with neutral atoms in an optical lattice*. *Fortschr. Phys.*, 48, 9-11, 925–943.
[10] Y. Dong, K. B. Whaley, and L. Lin. 2022. *A quantum hamiltonian simulation benchmark*. *npj Quantum Inf.*, 8, 1.
[11] R. France and B. Rumpe. 2007. *Model-driven development of complex software: a research roadmap*. *Proc. IEEE FOSE*, 37–54.
[12] M. Franz et al. 2024. *Co-design of quantum hardware and algorithms in nuclear and high energy physics*. *EPJ Web of Conferences*, 295, 12002.
[13] S. Fresca and A. Manzoni. 2021. *Real-time simulation of parameter-dependent fluid flows through deep learning-based reduced order models*. *Fluids*, 6, 7.
[14] D. Hangleiter, J. Carolan, and K. P. Y. Thébault. 2022. *Analogue Quantum Simulation: A New Instrument for Scientific Understanding*.
[15] P. Hauke et al. 2013. *Quantum simulation of a lattice schwinger model in a chain of trapped ions*. *PRX*, 3, 041018, 4.
[16] L. Lamata et al. 2018. *Digital-analog quantum simulations with superconducting circuits*. *Adv. Phys.: X*, 3, 1, 1457981.
[17] D. Lobser et al. 2023. *Jaqalpaw: a guide to defining pulses and waveforms for jaqal*. (2023).
[18] A. Lucas. 2014. *Ising formulations of many np problems*. *Front. in Phys.*, 2.
[19] Y. Meurice, R. Sakai, and J. Unmuth-Yockey. 2022. *Tensor lattice field theory for renormalization and quantum computing*. *RMP*, 94, 2.
[20] A. Miessen et al. 2022. *Quantum algorithms for quantum dynamics*. *Nature Computational Science*, 3, 1, 25–37.
[21] J. M. Murillo et al. 2025. *Quantum software engineering: roadmap and challenges ahead*. *ACM Trans. Softw. Eng. Methodol.*
[22] PASQAL developers. 2025. *PASQAL*. https://www.pasqal.com.
[23] Qiskit developers. 2025. *Qiskit: an open-source framework for quantum computing*. (2025).
[24] R. Ramsauer and W. Mauerer. 2025. *Towards system-level quantum-accelerator integration*. *Proc. IEEE QCE*.
[25] L. Schmidbauer and W. Mauerer. 2025. *Sat strikes back: parameter and path relations in quantum toolchains*. *Proc. IEEE QSW*, 01–12.
[26] L. Schmidbauer et al. 2024. *Polynomial reduction methods and their impact on qaoa circuits*. *Proc. IEEE QSW*.
[27] A. F. Shaw et al. 2020. *Quantum algorithms for simulating the lattice schwinger model*. *Quantum*, 4, 306.
[28] M. Strobl et al. 2025. *QML-Essentials–a framework for working with quantum fourier models*. *Proc. IEEE QSW*, 238–243.
[29] M. Troyer and U.-J. Wiese. 2005. *Computational complexity and fundamental limitations to fermionic quantum monte carlo simulations*. *PRL*, 94, 170201, 17.
[30] B. Yang et al. 2020. *Observation of gauge invariance in a 71-site bose–hubbard quantum simulator*. *Nature*, 587, 7834, 392–396.
[31] R. Au-Yeung et al. 2024. *Quantum algorithms for scientific computing*. *Rep. Prog. Phys.*, 87, 11, 116001.
[32] T. Yue et al. 2023. *Challenges and opportunities in quantum software architecture*, 1–23.
[33] Z.-Y. Zhou et al. 2022. *Thermalization dynamics of a gauge theory on a quantum simulator*. *Science*, 377, 6603, 311–314.