Differentiable Simulations for R&D: the path to an AI Physicist

Eliminating the simulation-reality gap through physics-based machine learning

Anurag Saha Roy & Shai Machnes

qruise

Contents

Abstract

Traditional physics-based simulations excel at modelling well-understood systems but often struggle with cutting-edge hardware R&D, where they are plagued by a simulation-reality gap that stems from our incomplete understanding of these novel technologies. This paper presents the idea of an AI Physicist, a novel approach combining differentiable physics simulations with machine learning to create digital twins that can learn from real-world data. By making physics equations differentiable, the system enables automated model learning, optimal control, and experiment design - capabilities that accelerate R&D cycles by orders of magnitude. We also demonstrate the approach through a case study in quantum computing, where the Qruise AI Physicist identified and resolved critical hardware issues in the operation of superconducting qubits in mere hours rather than several months. We then discuss a longer term roadmap for a general purpose AI Physicist.

1 Introduction

1.1 The Simulation-Reality Gap in Physics-centric R&D

Deep-tech R&D refers to technology development based on significant scientific or engineering breakthroughs, such as advanced AI, quantum computing, biotechnology, or novel materials. These companies are “deep” because they’re based on cutting-edge research that may have taken years to develop in labs or universities. *Physics-centric* R&D is a subset of deep-tech that emphasises ventures where the main challenge is the physics of the devices, e.g., low-field MRI, silicon photonics, or fusion reactors.

In physics-centric R&D, the standard workflow follows a predictable pattern: design, simulate, iterate until the predicted performance meets the design goals; then build, test, and if performance does not match expectation, debug the issues and iterate. Simulations tend to match reality for well-understood technologies, such as electric motors or low-speed airflow, but when R&D pushes beyond the state-of-the-art into the development of new technologies, our understanding of the device is often incomplete. We face physical effects not yet captured by existing models, larger-than-expected manufacturing variations, and sensitivities to operational environments that aren’t fully accounted for, among other challenges.

These gaps in understanding translate into a simulation-reality gap, resulting in devices whose performance fails to meet simulation benchmarks. Eliminating this gap requires deeper insight into the performance bottlenecks of these devices and is key to advancing physics-centric technologies, such as quantum computing and photonics.

Traditional simulation software, such as the widely used tools from Ansys, Dassault, or MathWorks, excel at modelling well-understood phenomena, but fall short when dealing with:

- Novel physics-centric technologies such as neuromorphic computing or silicon photonics
- Systems with significant unknown parameters typical of new technologies where the fabrication processes are often not reproducible or sufficiently mature
- Determination of physics models from real-world data, a.k.a. the *Inverse Problem*. These

data-to-model workflows are critical for developing new technologies, as they help close the simulation-reality gap and uncover missing elements in first-principle models

- Real-time adaptation based on device feedback, which is key to the design-fabricate-test-redesign iteration cycle

1.2 The Emergence of Differentiable Programming

Recent advances in machine learning have introduced differentiable programming, a paradigm that treats entire programs as mathematical functions whose gradients can be computed automatically. This breakthrough enables the same learning techniques that revolutionised neural networks to be applied to physics simulations, unlocking unprecedented capabilities for data-driven system identification and optimisation.

2 Differentiable simulations: core technology

2.1 Mathematical Foundation

A simulation can be viewed as a mathematical function $f(x_1, x_2, \ldots, x_n)$ that maps model and control parameters to observed (measured) values.

In traditional physics-based modelling, the simulation will most often include solving differential equations, such as an equation of motion or a state equation. The process of solving these equations will normally include discretisation, which replaces the gradients in time and space within the differential equations with finite differences between discrete points. Discretisation of the time axis forms the basis of time-propagation methods such as Runge-Kutta. Discretisation of spatial dimensions is often termed finite element modelling or finite volume modelling, depending on the details.

We now consider the partial derivatives of the simulation function, $f(x_1, x_2, \ldots, x_n)$, with respect to the model and control parameters: $\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_n}$. *Automatic differentiation* (AD) [1] is a technique for precise and efficient evaluation of derivatives for functions implemented as computer programs. A *differentiable simulation* computes both f and $\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_n}$ simultaneously by ensuring that each computational step used to compute f also produces the corresponding gradient information. This is implemented through a computational graph framework as explained in detail in Fig. 1.

Prior to the advent of AD, creation of differentiable simulations was a daunting, but not impossible, task. For example, one could construct a bespoke method for a differentiable simulation of the Schrödinger equation for discrete quantum-mechanical systems for the purposes of optimal control, a method known as GRAPE [2]. We note that the differentiability of the physics simulation as a whole (as enabled by AD) is a software feature, i.e., it allows programmatic evaluation of gradients. This concept is distinct from the differentiability inherent in physics simulations, which are often based on differential equations.

AD enabled a much more flexible and robust approach to differentiable physics simulations [3–5], along with powerful optimisation algorithms that efficiently explore high-dimensional parameter

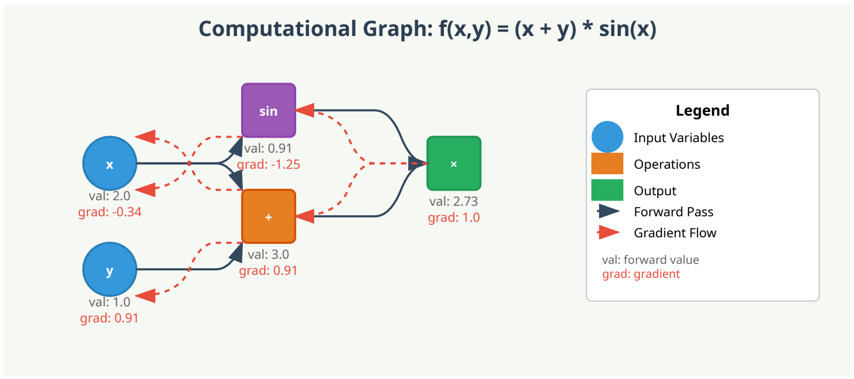

Figure 1: Computational graph for automatic differentiation of the function $f(x, y) = (x + y) \times \sin(x)$ demonstrating the AD forward and backward passes. The AD process operates through a two-phase computation on the directed acyclic graph structure. During the forward pass, function values are computed by following the directed graph from input nodes through intermediate operations to the final output, with each node storing its computed value for later use. The backward pass then computes gradients using the chain rule by flowing backward through the graph, where each node calculates and accumulates gradients from all outgoing paths. Each node in the computational graph stores both its forward-computed value and its backward-computed gradient, enabling simultaneous computation of function values and derivatives. Gradients accumulate at nodes with multiple incoming paths, properly implementing the chain rule's summation requirement for partial derivatives in complex computational expressions. The graph here shows input variables $x = 2.0$ and $y = 1.0$ (blue circles) flowing through intermediate operations: addition (+, orange rounded rectangle), sine function (sin, purple rounded rectangle), and multiplication (×, green rounded rectangle). Each node displays both its forward-computed value (val) and backward-computed gradient (grad). Solid black arrows indicate the forward pass direction where function values are computed following the directed acyclic graph structure. Dashed red arrows show the backward pass where gradients flow in reverse direction using the chain rule of differentiation.

spaces to fit models to experimental data. Deep learning algorithms are, under-the-hood, efficient optimisation routines. AD brings this same capability — parameter optimisation and learnability — to physics-based system models. See Fig. 2 for more details.

In the next sections, we briefly review how automatic differentiation (AD) compares to more traditional techniques such as symbolic and numerical differentiation, offering unique advantages over both approaches.

Symbolic Differentiation

Symbolic differentiation manipulates mathematical expressions algebraically to produce the exact derivative formulas. For example, given $f(x) = x^2 + \sin(x)$, it would produce $f'(x) = 2x + \cos(x)$ as a new symbolic expression.

Advantages: Produces exact results with no numerical error, and the derivative formula can be reused for any input values.

Disadvantages: Expression size swells exponentially larger as derivatives of complex functions are often more complex than the original. Symbolic differentiation also struggles with conditional statements, loops, and other programming constructs that don't translate well to symbolic mathematics.

Finite Difference Methods

Finite differences (FD) approximate derivatives using

$$f'(x) \approx \frac{f(x+h) - f(x)}{h}$$

for small h or similar definitions.

Advantages: Simple to implement and works with any black-box function.

Disadvantages: Inherent approximation error that creates a trade-off: larger h gives truncation error, smaller h gives round-off error due to floating-point precision limits in the calculation of f. Computing gradients of functions with many inputs requires many function evaluations (at least $n + 1$ for n variables), making it expensive for high-dimensional problems.

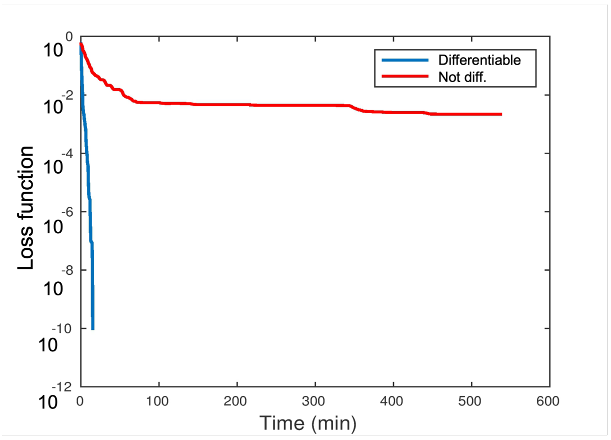

Figure 2: Any good learning algorithm is, at its core, an optimisation process minimising a function that measures the distance between what is and what should be, typically termed a *loss function*. Here, we show how quickly a loss function can be minimised depending on whether the learning/optimisation process uses a differentiable simulation. In this example, the loss function depicts the predicted error rate of a quantum gate. The x-axis indicates the running time of the learning process. The speed and accuracy of the optimisation is denoted for a differentiable (blue) vs a non-differentiable (red) simulation. It's clear that differentiability offers several orders of magnitude improvement for both metrics, demonstrating how learning without differentiability is practically infeasible. Figure adapted from [4].

Automatic Differentiation

AD computes exact derivatives by applying the chain rule systematically during program execution. It treats programs as sequences of elementary operations (addition, multiplication, sin, cos, etc.) and accumulates derivatives alongside the original computation (see Fig. 1 for details).

Forward mode AD: Propagates derivatives in the same direction as the original computation. Efficient when you have few inputs and many outputs.

Reverse mode AD (back-propagation): Propagates derivatives backward through the computation graph. Efficient when you have many inputs and few outputs, which is why it's dominant in machine learning.

Key advantages of AD:

- Produces machine-precision accurate derivatives (no approximation error)
- Computational cost is only a small constant factor compared to evaluation of the original function
- Handles arbitrary program control flow, conditionals, and loops
- No expression swell issues

Disadvantages: Requires either source code transformation or operator overloading, and can have higher memory requirements (especially reverse mode) due to storing intermediate gradient values.

2.2 Implementation Challenges for Differentiable Physics Simulation

Creating differentiable physics simulations requires fundamental changes to the simulation engine architecture:

1. **Computational graph design**: Simulations must be structured as computational graphs similar to neural networks.
2. **Automatic differentiation**: All mathematical operations must support gradient computation.
3. **ML software stack integration**: Must be implemented using frameworks like TensorFlow or JAX [6] for AD.
4. **Numerical stability**: Ensuring gradients remain well-behaved through complex physics calculations.

Critically, **differentiability cannot be retrofitted** into existing simulation engines; it must be established as a core design principle from the outset.

2.3 Comparison with Alternative Approaches

Most existing R&D simulations are typically one of the following two types:

- Physical modelling based on well-explained phenomena and highly accurate dynamics. These are computationally heavy but great for traditional engineering applications such as automotives or electric motors. However, such simulators are incapable of learning from real-world data to fill gaps in first-principles modelling.
- Neural-network-based simulators that are trained on physics simulations (shifting computational costs to training time) to produce approximately accurate dynamics (amortised computation) and can learn from data. These surrogate models are great for fast, approximate solutions, but they lack any explainability or insights into underlying physical phenomena. Moreover, the learned behaviour cannot be back-propagated from neural networks to the physics simulation, e.g., for optimising parameters.

Simulation type	Accuracy	Learn from data?	Physical insight?	Computational cost
Legacy physics	Very high	×	Perfect	High
Neural networks	Low-medium	✓	None	Low
Differentiable physics	**Very high**	✓	**Perfect**	**Medium**

Table 1: Comparison of simulation approaches

The unique strength of differentiable physics simulations stems from the combination of the accuracy and interpretability of physics-based models with the learning capabilities of machine learning systems. This is summarised in Table 1. It's important to note here that methods such as physics informed neural networks (PINN) [7], which promise to provide explainability to neural-network-based models are typically extremely brittle, unreliable and finicky in practice and often do not work for any form of complex physics models of practical relevance [8–11].

3 What can we do with differentiable simulations?

An AI Physicist built on top of differentiable simulations delivers unique and significant value to organisations developing physics-centric technologies. The combination of first-principles simulations with machine learning through differentiable programming bridges the simulation-reality gap and allows unprecedented speed, accuracy, and intelligence in the R&D process [12–17].

3.1 Performance Advantages

Experimental results demonstrate that differentiable optimisation provides dramatic speed improvements over gradient-free methods [18–20]. Fig. 2 discusses a concrete example in design of controls for quantum computing where the differentiable approach shows convergence in ∼50 iterations while the non-differentiable approach typically plateaus at a non-optimal solution after 500+ iterations with 1000× slower convergence. The performance advantage of differentiable simulations is required to make complex multi-parameter optimisation practical, allowing for realistic system models with hundreds and thousands of adjustable parameters. To make these enhancements possible, the AI Physicist fundamentally changes the R&D workflow:

Traditional Cycle: Design → Simulate → Build → Test → Debug → Redesign (months per iteration)

AI physicist cycle: Design → Simulate → Build → Test → **Auto-debug** → **Auto-optimise** (weeks per iteration)

The automated debug and optimisation phases, powered by differentiable simulations, eliminate the most time-consuming and expertise-dependent steps.

3.2 Scientific and Technical Value

Differentiable simulations enable automated system identification and real-time closed-loop optimisation, thus creating accurate, evolving digital twins that offer deep insight into device behaviour and performance bottlenecks, as explained below:

- **Automated system identification**: Automatically extracts governing physics parameters from sparse, noisy experimental data by solving the *Inverse Problem* without requiring hand-crafted experiments.
- **Closed-loop optimisation**: Enables real-time interaction with hardware by executing experiments, refining models, and computing optimal control solutions in a continuous feedback loop.
- **True digital twins**: Builds accurate, interpretable, physics-grounded digital twins that evolve with the system and provide insight into performance-limiting effects such as decoherence, noise, and signal distortion.
- **Differentiable everything**: A fully differentiable stack enables gradient-based optimisation across hundreds of parameters with orders-of-magnitude acceleration compared to traditional gradient-free methods.

3.3 Business and Operational Value

Beyond advancing scientific understanding, differentiable simulation powered AI Physicists promise a significant impact on both the overall R&D effort and the value delivered to host organisations:

- 10–$100\times$ **faster debug and calibration cycles**: What previously required months of PhD-level effort can now be executed in hours or days (see section 4.4). This drastically reduces downtime in high-cost experimental setups (e.g., dilution refrigerators, MRI magnets).
- **Accelerated time-to-market**: Reduced iteration times directly translate to faster development cycles, critical in competitive industries like quantum computing, photonics, and medical imaging.
- **Reduced dependence on scarce expertise**: The AI Physicist captures and automates domain-specific knowledge, enabling a small team of highly trained physicists and engineers to accomplish work that would previously have required significantly larger teams with equally rare expertise.
- **Improved yield and performance**: Differentiable learning uncovers subtle failure modes and performance bottlenecks, enabling systematic corrections that improve fidelity, efficiency, and manufacturing yield.
- **Scalable across domains**: The framework generalises beyond quantum to MRI, NMR, EPR, silicon photonics, and other fields with complex physics and limited data. As long as the physics is simulable, it is optimisable.

Accelerated physics-centric R&D enables faster development and deployment of transformative technologies - from quantum computing to advanced medical imaging - directly impacting healthcare, energy, and information systems. By reducing time-to-discovery and lowering barriers to innovation, it empowers humanity to address complex global challenges with unprecedented speed and precision.

4 The Qruise AI Physicist for quantum technologies

We now discuss a concrete implementation of a differentiable simulation based AI Physicist developed by Qruise with applications in quantum computing and sensing. We outline the broad architecture and briefly discuss the technical implementation before diving into an example use-case of debugging operations in superconducting quantum computers.

4.1 System Overview

The Qruise AI Physicist operates as an autonomous research assistant that interacts directly with hardware in real-time. It uses QruiseOS to plan and execute experiments that gather informative data regarding the device under development. It then uses QruiseML to update comprehensive physics models based on these experimental results and provides actionable insights for system improvement. The software can also generate optimal control strategies for enhanced operation of the device. This creates a closed-loop system where the digital twin continuously improves its accuracy to match the physical device, and utilises this information to improve device performance. The workflow of the Qruise AI Physicist is depicted in Fig. 3.

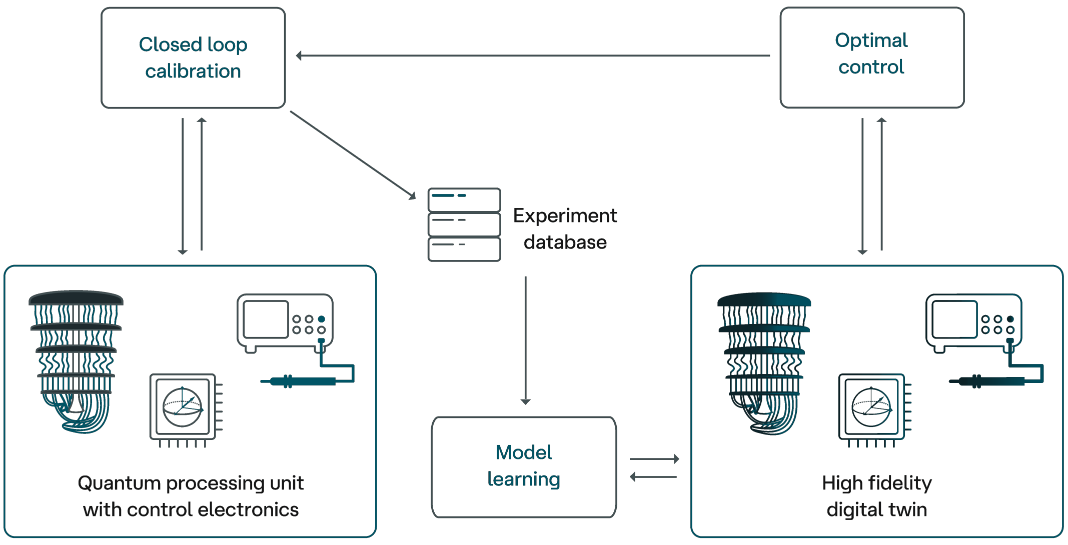
Closed loop calibration
Optimal control
Experiment database
Quantum processing unit with control electronics
Model learning
High fidelity digital twin

Figure 3: Architecture of the Qruise AI Physicist. While the graphical elements suggest use for quantum technologies, the architecture and most components do not change when applied to other verticals, with the main difference being the differential equations implemented inside the digital twin.

4.2 Technical Implementation

In this section we briefly discuss the technical implementation underpinning the Qruise AI Physicist architecture described in Fig. 3.

4.2.1 Software Stack

The AI Physicist is built on modern Machine Learning software infrastructure such as TensorFlow and JAX (and more recently Julia) to enable automatic differentiation. Custom differentiable solvers have been developed for domain-specific use-cases. Additionally, the software is interfaced to the hardware to allow real-time control of measurement systems. An extensive library of optimisation algorithms, both gradient-based and (gradient-free with gradient-based) methods, allows efficient exploration of design and parameter landscapes for a multitude of applications. All of this is aided by a comprehensive visualisation tool that allows interactive exploration of the learned models.

4.2.2 Integration Requirements

For successful deployment, the software must be able to interface with the control hardware to enable connection to the experimental apparatus. High-throughput data collection is supported through joint solutions developed with control electronics manufacturers. Some domain knowledge of the relevant physics is needed to set up the initial baseline model. Since the algorithms are physics-based, they can work with small sparse data but often benefit from the use of modern computational accelerators such as GPUs for large-scale simulation and optimisation.

4.2.3 Deployment Modes

The software supports multiple deployment configurations:

Cloud-based	For initial prototyping and non-sensitive applications
On-premise	For security-critical applications and proprietary hardware
Hybrid	For cases where a combination of cloud-based computation and local data processing is preferred

4.3 Enabled Capabilities

The Qruise AI Physicist unlocks and enhances a host of capabilities for AI-driven science and engineering:

- **Model learning:** Automatically adjusts simulation parameters to match experimental observations, closing the simulation-reality gap through data-driven parameter estimation.

- **Model-based control:** Learns optimal control sequences to achieve desired device behaviour, accounting for all known physics and learned system characteristics.
- **Experiment design:** Determines which experiments will provide maximum information to improve the model, optimising the data collection process.
- **Error budget analysis:** Quantifies how much each physical phenomenon limits device performance, enabling predictive insights into limitations and failure modes.

The above capabilities are, in fact, optimisation problems. For example, model learning minimises the difference between simulation predictions and real-world data. None of these would be practical without the differentiable simulations that allow for efficient gradient-based optimisation.

4.4 Case Study: Debugging Quantum Computing Hardware

4.4.1 Problem Description

Dr Lior Ella, Director of Research at Quantum Machines, encountered misbehaving qubits in a QuantWare superconducting quantum processing unit (QPU). The misbehaviour could be broadly summarised as the quantum instructions, i.e., 2-qubit gates, not performing up to the expected performance levels (as predicted by their calculations) in operational systems. Traditional debugging approaches would require extensive manual analysis, parameter sweeps, and expert interpretation, a process that typically takes months and that had so far not provided any concrete insights.

4.4.2 AI Physicist Intervention

In order to tackle the above problem, the Qruise AI Physicist was deployed with:

- **Input**: Existing experimental data from the QPU (one and two qubit measurements)
- **Goal**: Identify the root cause and provide actionable solutions
- **Constraints**: Make do with available data - no additional device time allocated

4.4.3 Results

You can see the results generated by the Qruise AI Physicist in Fig. 4. Within hours, it:

1. **Learned 83 system parameters** of a two-qubit model from the available data.
2. **Identified the root cause**: Signal distortion as the signal goes from room temperature control electronics and through the cryogenic fridge to the superconducting QPU.
3. **Provided quantitative analysis** of the distortion characteristics.
4. **Suggested corrective measures** that could be immediately implemented.

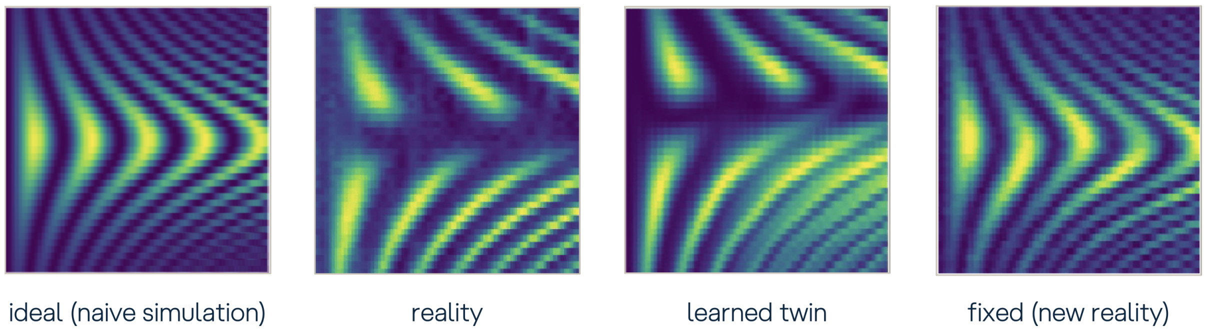

Figure 4: Learning and fixing errors in 2-qubit gates for superconducting qubits. The steps are, left to right: (a) A naive simulation of the idealised system where nothing is going wrong. (b) Actual data from the device - clearly the system is behaving in a way which is very different than expected. (c) In order to fix the problem, we must understand what is causing the misbehaviour. In other words, we need the model, i.e. the physics equations, that reproduce the process. We apply Qruise's model learning algorithms to identify the model that replicates the observed behaviour. (d) Examining the learned physics model, it becomes clear the issue is distortion of one of the control signals as it goes from room temperature to the superconducting qubits at 0.03 K - an unavoidable distortion, but one we can easily compensate for, resulting in behaviour which is very close to the ideal.

To produce the above results, a differentiable model of the whole stack including the QPU and the control electronics was first developed using the tools in QruiseML. Once this digital twin was ready, it was fed data from previously conducted experiments and the model learning algorithm described in Fig. 5 was allowed to improve the digital twin based on this experimental data. It could then correctly predict the previously mischaracterised system parameters and also offer insights into how to fix them. In a matter of few hours, the system accomplished what would typically require months of expert physicist time.

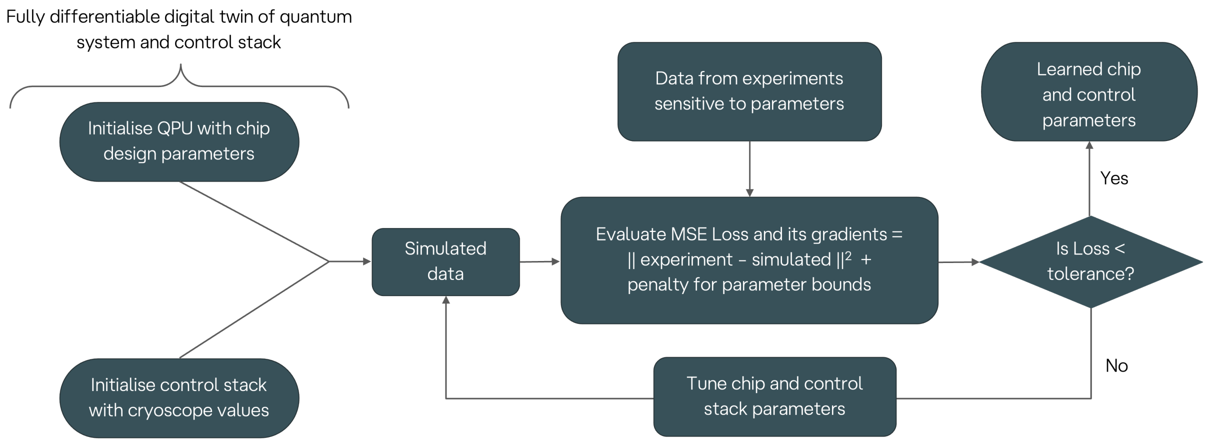
Fully differentiable digital twin of quantum system and control stack
Initialise QPU with chip design parameters
Initialise control stack with cryoscope values
Simulated data
Data from experiments sensitive to parameters
Evaluate MSE Loss and its gradients = || experiment - simulated ||² + penalty for parameter bounds
Is Loss < tolerance?
Yes
Learned chip and control parameters
No
Tune chip and control stack parameters

Figure 5: Block diagram of model learning for debugging quantum computing hardware. At its core, the model learning process minimises the distance between experimental data and simulated results by optimising over the space of model parameters. To initialise the optimisation process, we start with a model that includes only the minimal information already known about the system. The loss function for the optimisation is the mean squared error (MSE) between the experimental data (denoised and rescaled) and the results of the simulation of identical experiments. The optimisation utilises efficient gradient-based optimisation algorithms (e.g., Adam), as the Qruise digital twin implements a differentiable simulation framework. The optimisation stops when the loss function drops below the required tolerance.

The differentiable model of the complete stack includes the following components:

- Qubit physics (Hamiltonian & Lindbladian dynamics): frequencies, anharmonicities, couplings, decay and dephasing effects, crosstalk
- Control & measurement chain characteristics: effects from non-ideal behaviour of electronic components such as analogue-to-digital converters, filters, and arbitrary waveform generators
- Signal distortions in cryogenics: deformations introduced by cables carrying microwave pulses to the QPU inside the cryogenics
- Environmental noise sources: Markovian and non-Markovian noise originating either from the QPU or the classical control stack

By fitting this comprehensive model to experimental data, the system could isolate the specific contribution of each component to the observed behaviour. For a more in-depth explanation of the model learning algorithm, please refer to [21].

5 General purpose AI Physicist — the long term vision

The Qruise AI Physicist presented here can be viewed as a junior PhD student specialising in quantum technologies. And while an infinite supply of junior physicists working 24/7 is of tremendous value to any R&D effort, we're still just scratching the surface. Going forward, we expect progress along three major axes: increased scientific maturity, increased technical capabilities, and expanding domain expertise.

5.1 Increased scientific maturity

As the AI Physicist progresses from a junior PhD student, to a senior PhD student, post-doc and eventually senior scientist, we will see gain of functional capabilities. For physicists, the natural way to represent equations is using symbolic representation. AI Physicists will support direct conversion from symbolic to numeric models. And when user-supplied models fall short, scientific literature will be automatically used to fill-in the missing pieces, as well as perform automated model discovery by learning physical relationships from the data (c.f. SINDy [22]). Model discovery will be made more efficient by incorporating Bayesian considerations in automated data acquisition and model learning. Model ambiguities, where more than one model fits the data, will be automatically resolved, and any fear of over-fitting will be similarly tested and resolved.

The end goal is to implement a fully functional AI Physicist capable of executing the entire scientific process: review literature → create hypothesis → model physics → simulate behaviour → design prototype → fabricate & test → search for missing phenomena → remodel to fill gaps → repeat until convergence. When given adequate literature to learn from and adequate tools to execute the steps above, the AI Physicist should be able to do this for any new field.

5.2 Increased technical capabilities

As the technology expands to new domains, key challenges include:

- **Model complexity**: Balancing detail with computational tractability.
- **Multi-physics coupling**: Handling of interactions between different physical phenomena.
- **Uncertainty quantification**: Providing reliable confidence estimates.
- **Real-time constraints**: Meeting time requirements for closed-loop control.

The intersection of physics and machine learning is a highly active field with ongoing developments in various directions. Multi-fidelity modelling, which combines high- and low-fidelity simulations based on the required level of accuracy, is critical for exploring large design spaces with thousands of parameters. As mentioned above, uncertainty quantification remains a key challenge and some researchers are addressing this through the development of robust optimisation methods that provide guarantees regarding the model's resilience to manufacturing variations and parameter inhomogeneities. There is also a lot of focus on generalising these tools through transfer learning techniques.

5.3 Expanding domain expertise

When expanding the capabilities of the AI Physicist from one domain to the next, it's often useful to move into adjacent fields that share a lot of the underlying physical equations for modelling, use the same kind of equipment for controlling & driving the systems, or have the same kind of noise & error generating phenomena. For example, starting with quantum computing and quantum sensing would make nuclear magnetic resonance (NMR) and magnetic resonance imaging (MRI) a natural next step since they share the same quantum mechanical equations used for modelling system dynamics. Similarly, the photonic devices used in quantum computers share the same kind of miscalibration errors that occur in other silicon photonics-based technologies.

The success of differentiable physics simulations could thus transform multiple industries by accelerating innovation cycles in several physics R&D sectors [12–17]. Reducing development costs through automation is not only a financial incentive, but also a strategic necessity to enable faster time-to-market of translational technologies. It can also enable the productisation of completely new technologies previously limited by modelling capabilities.

Eventually, in every lab developing physics-centric technology, from nano-mechanical systems to fusion reactors, you'll find an AI Physicist working alongside human physicists and engineers, accelerating our civilisation towards a better future.

6 Conclusions

An AI Physicist represents a fundamental advancement in simulation technology by uniting first-principles physics with machine learning through differentiable programming. It eliminates the longstanding simulation-reality gap that has plagued R&D in emerging technologies, enabling automated model learning, optimal control, and experiment design directly from real data. This transforms the traditional R&D workflow from manual, expert-driven iterations to an automated, data-driven, closed-loop cycle.

Our case study in quantum computing illustrates the immediate practical value of this approach: debugging and optimising a highly complex and sensitive device — a task that would normally require months of expert effort — was accomplished in mere hours. By creating fully differentiable digital twins that capture the entire system — from device physics to the control stack — the AI Physicist not only diagnosed the source of malfunction, but also suggested and validated corrective measures using only existing data.

The performance gains enabled by differentiable simulations are profound. Gradient-based optimisation converges orders of magnitude faster than conventional methods, making it feasible to work with complex, high-dimensional system models. These capabilities unlock unprecedented speed and precision across a range of physics-centric fields, including quantum technologies, MRI and silicon photonics. As the technology matures and expands to new domains, it has the potential to accelerate innovation across all physics-centric deep technology industries.

Compared to legacy simulation platforms and emerging ML-enhanced tools, the AI Physicist framework described here will require a simulation engine that is differentiable by design, vertically integrated with hardware, and built to support the full data-to-model-to-control workflow. As R&D increasingly shifts toward deep-tech domains where physics is the bottleneck, this offers both a scientific and strategic advantage. By accelerating discovery, reducing requirements of scarce human expertise, and enabling rapid iteration in systems once deemed intractable, AI Physicists will stand alongside their human counterparts, amplifying their creativity and accelerating collective progress toward a better future.

References

[1] A. Griewank and A. Walther. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. SIAM, 2008.

[2] N Khaneja et al. "Optimal Control of Coupled Spin Dynamics: Design of NMR Pulse Sequences by Gradient Ascent Algorithms". In: *Journal of Magnetic Resonance* 172.2 (2005).

[3] K. Jatavallabhula et al. "gradSim: Differentiable simulation for system identification and visuomotor control". In: *International Conference on Learning Representations*. 2021.

[4] S. Machnes et al. "Tunable, Flexible, and Efficient Optimization of Control Pulses for Practical Qubits". In: *Physical Review Letters* 120 (2018).

[5] C. Rackauckas et al. *Universal Differential Equations for Scientific Machine Learning*. 2022. arXiv: 2205.14249.

[6] J. Bradbury et al. *JAX: composable transformations of Python+NumPy programs*. 2018. URL: http://github.com/jax-ml/jax.

[7] M. Raissi et al. "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations". In: *Journal of Computational Physics* 378 (2019).

[8] P. Chuang and L. Barba. *Experience report of physics-informed neural networks in fluid simulations: pitfalls and frustration*. 2022. arXiv: 2205.14249.

[9] A. Krishnapriyan et al. “Characterizing possible failure modes in physics-informed neural networks”. In: *Advances in Neural Information Processing Systems*. Vol. 34. 2021.

[10] T. Grossmann et al. “Can physics-informed neural networks beat the finite element method?” In: *IMA Journal of Applied Mathematics* 89.1 (2024).

[11] P. Chuang and L. Barba. *Predictive Limitations of Physics-Informed Neural Networks in Vortex Shedding*. 2023. arXiv: `2306.00230`.

[12] J. Citrin et al. *TORAX: A Fast and Differentiable Tokamak Transport Simulator in JAX*. 2024. arXiv: `2406.06718`.

[13] A. Franz et al. *PICT – A Differentiable, GPU-Accelerated Multi-Block PISO Solver for Simulation-Coupled Learning Tasks in Fluid Dynamics*. 2025. arXiv: `2505.16992`.

[14] C. Panuski et al. “A full degree-of-freedom spatiotemporal light modulator”. In: *Nature Photonics* 16 (2022).

[15] T. Dorigo et al. “Toward the end-to-end optimization of particle physics instruments with differentiable programming”. In: *Reviews in Physics* 10 (2023).

[16] T. Xue et al. “JAX-FEM: A differentiable GPU-accelerated 3D finite element solver for automatic inverse design and mechanistic data science”. In: *Computer Physics Communications* (2023).

[17] S. Mann et al. “∂PV: An end-to-end differentiable solar-cell simulator”. In: *Computer Physics Communications* 272 (2022).

[18] R. Antonova et al. “Rethinking Optimization with Differentiable Simulation from a Global Perspective”. In: *Conference on Robot Learning (CoRL)*. 2022.

[19] C. D. Freeman et al. *Brax: A Differentiable Physics Engine for Large Scale Rigid Body Simulation*. 2021. arXiv: `2106.13281`.

[20] H.J. T. Suh et al. “Do Differentiable Simulators Give Better Policy Gradients?” In: *Proceedings of the 39th International Conference on Machine Learning (ICML)*. 2022.

[21] N. Wittler et al. “Integrated tool-set for Control, Calibration and Characterization of quantum devices applied to superconducting qubits”. In: *Physical Review Applied* 15 (2021).

[22] S. Brunton et al. “Discovering governing equations from data by sparse identification of nonlinear dynamical systems”. In: *Proceedings of the National Academy of Sciences* 113.15 (2016).