Dark Matter Simulations with Quantum Physics-Informed Neural Networks (QPINNs)

GUILLAT Arthur
arthur.guillat@epfl.ch

September 30, 2025

Contents

Abstract

This report explores an innovative approach for making dark matter simulations, specifically in the case of the fuzzy model. The idea is to train a neural network on a quantum computer while specifically enforcing the laws of physics in order. We call the result a *Quantum Physics-Informed Neural Network* (QPINN). The goal here is practical: learn solutions to partial differential equations (PDEs) that appear in astrophysics with as little data as possible while respecting the underlying physics (by construction). We focus on three test cases of increasing difficulty: the one-dimensional Poisson equation, the heat equation, and the coupled Schrödinger–Poisson system. The latter case models *fuzzy dark matter* (ultra-light particles behaving as waves on galactic scales). We start by introducing the needed background (PINNs, quantum circuits, and fuzzy dark matter), describe how the model is built, and show what worked, what did not, and what we can hope to do from there.

1 Introduction

Almost a century ago, astronomers realized that galaxies rotate too fast to be held together by the gravity of visible stars and gas alone. Fritz Zwicky proposed that an invisible component, which know as dark matter, must supply extra mass. Today, evidence from galaxy rotation or gravitational lensing suggests that dark matter makes up most of the matter in the universe. Despite its overwhelming presence, it has - to this day - never been detected. Therefore, astrophysicists make up models to attempt describing its properties. Those models mostly differ by how they make the early universe look, in particular regarding early galaxies formation. The figure below (1) illustrates the difference between three of those models.

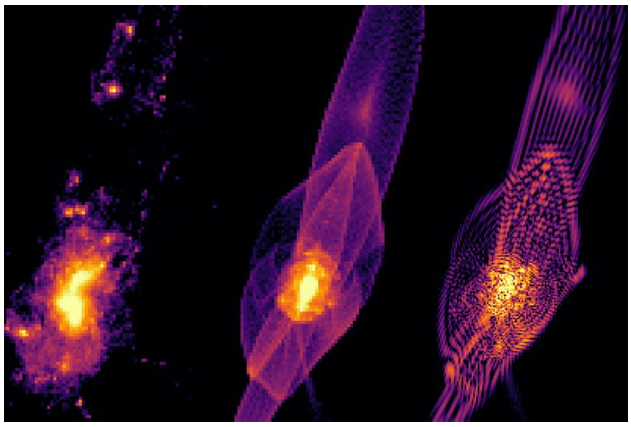


Figure 1: Early galaxies forming according to different DM models. In order (left to right): cold, warm and fuzzy dark matter

1. Cold dark matter: made of heavy, slow-moving particles that clump together strongly, resulting in bright halos

2. Warm dark matter: made of lighter and faster particle than the previous model. As a consequence, it doesn't form the same dense regions but instead creates filament like structures.

3. Fuzzy dark matter: made of ultra light particles ($10^{-22}eV$). While it looks similar to the warm dark matter case, its wavefunction stretches on the scale of galaxies. It thus behaves more like a wave than a particle, creating interference patterns all the way through the galaxy formation.

For this project, we were interested in the last case. Unfortunately, it is numerically difficult to simulate, and for two reasons: First, wave equations need high resolution to resolve interference. Second, the coupling to gravity is nonlinear. At the same time, data for

training black-box machine learning models are limited or expensive to generate. Physics-Informed Neural Networks (PINNs) offer a way around the data issue: instead of training on labeled examples, we penalize violations of the governing PDEs at *collocation points*. This way, the model relies on physical laws instead of big datasets.

The present work goes one step further. Instead of a standard neural network, we use a small quantum circuit as the function approximator. The intent is not to claim speedups out of reach for classical hardware today. It is to test whether quantum circuits are interesting candidates for quantum simulations when trained with physics-informed losses. We use Strawberry Fields to build circuits in the continuous-variable (CV) setting, where optical modes ("qumodes") carry the information and gates such as displacement, rotation, squeezing, beam splitters and Kerr operations act on them. The combination of CV circuits with PINN training forms what we call a QPINN.

The paper is organized as follows. Section 2 finishes to introduce the key concepts (PINNs and basic quantum computing ideas). Section 3 describes the QPINN design: inputs, layers, measurement, and the loss. Section 4 reports results on Poisson, heat, and Schrödinger–Poisson. We conclude with limitations and a realistic outlook.

2 Background

2.1 Physics-Informed Neural Networks (PINNs)

Differential equations frame how fields evolve and interact. Analytical solutions (exact solutions) are rare, and numerical solvers trade accuracy for time and memory. PINNs sit in between. The network u_θ is trained so that the *residual* of the PDE (deviation from physical laws) is small at collocation points and the boundary/initial conditions are honored. A generic loss has the form

$$L(\theta) = \lambda_{\text{PDE}} \frac{1}{N} \sum_{i=1}^{N} \|\mathcal{N}[u_\theta](x_i) - f(x_i)\|^2 + \lambda_{\text{BC}} \frac{1}{N_{\partial\Omega}} \sum_{j} \|\mathcal{B}[u_\theta](x_j) - g(x_j)\|^2 + \lambda_{\text{IC}} \cdots , \quad (1)$$

where $\mathcal{N}$ is the differential operator, $\mathcal{B}$ encodes boundary conditions, and the λ's are weights used to balance the terms. Derivatives are computed by automatic differentiation (AD), which avoids finite-difference noise and simplifies nested derivatives such as Laplacians.

2.2 Quantum computing in a nutshell

We now present some of the fundamental quantum computing concepts. Unlike a classical bit, which can only take the values 0 or 1, a quantum state can exist in a *superposition*

of basis states. Each basis state is assigned a complex amplitude, and these amplitudes not only have magnitudes (which determine probabilities when measured) but also phases (which determine how different states interfere). When we perform a measurement, the superposition collapses to one of the basis states, with a probability equal to the squared magnitude of its amplitude.

In the *continuous-variable* (CV) model of quantum computing, the underlying basis is not just a pair of discrete states, but a full continuum, typically expressed in terms of position and momentum variables. This framework is especially natural in quantum optics, where information is carried by light modes. In this case, operations on the quantum state are implemented by optical components acting as gates: for example, displacements, rotations, squeezing, beam splitters, or Kerr nonlinearities. These gates transform the state in phase space, and by combining them we can build expressive quantum circuits that serve as the backbone of continuous-variable quantum computation. Five gates are enough for our purposes:

- **Displacement** $D(\alpha)$ shifts the mode in phase space.
- **Rotation** $R(\phi)$ rotates the quadratures.
- **Squeezing** $S(r)$ rescales uncertainties along conjugate axes.
- **Beam splitter** $BS(\theta)$ mixes two modes (a rotation in mode space).
- **Kerr** $K(\kappa) = e^{i\kappa(a^\dagger a)^2}$ adds non-Gaussianity (nonlinear phase).

An interferometer is built from beam splitters and phase shifters. Contrarily to classical computers, in which you can access the information "directly", a quantum computer requires a measurement. As previously mentioned, this collapses the superposition and the measurement result (which is the program output) is random. One of the key aspects of making quantum algorithms is to find ways to increase the probability of the result you desire. We measure quadratures (e.g., $\hat{x}$) to read out an expectation value, average over shots if needed, and treat the result as the network output.

2.3 Fuzzy dark matter (FDM)

FDM treats dark matter as an ultra-light scalar field. At galactic scales, the non-relativistic limit gives a Schrödinger equation with a gravitational potential U that satisfies the Poisson equation. If Ψ is the wavefunction, the density is $\rho \propto |\Psi|^2$. The coupled system is

$$i\hbar\,\partial_t\Psi = -\frac{\hbar^2}{2m}\nabla^2\Psi + mU\,\Psi, \qquad \nabla^2 U = 4\pi G\left(|\Psi|^2 - \rho_0\right). \tag{2}$$

The background density ρ_0 sets the zero of potential. The result is a wave that can exhibit phenomena absent from cold dark matter. Simulating this system in 3D with high resolution is expensive; a compact surrogate that respects wave physics is attractive.

3 Quantum Physics-Informed Neural Networks

Similarly to classical computers, in which bits are acted upon through logic gates (AND, NOT, XOR... gates), quantum computing makes use of quantum gates. Those gates can be physically implemented as optical devices for example, and using a sequence of them forms a CV quantum program. In order to create a full CV quantum computing neural network, we make use of five quantum gates:

Finally, by using both beam splitter and rotation gates, we can construct an interferometer gate, which will be very useful for constructing a quantum neural network.
Classically, if we had n neurons at one layer, represented by $x \in \mathbb{R}^n$, then the m neurons at the next one would be given by

$$\mathcal{L}(x) = \phi(Wx + b) \tag{3}$$

where $W \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$ and ϕ is a nonlinear activation function. We can define a CV quantum neural network layer as $\Phi \circ D \circ U_2 \circ S \circ U_1$ - basically a sequence of five quantum gates. The parameters of those gates (angles, scaling factors...) are the weights which we hope to train using the network.

Each of those gates "acts" as one of the necessary steps to perform the operations described by (3): the first three gates (the two interferometer and squeezing gates) act as the matrix multiplication of W and x, whereas the displacement and Kerr gates are respectively similar to the addition of biases and the composition with an activation function. Depending on the number of qumodes we use, the precise disposition of gates may change, but the architecture stays constant. The figure (2) below illustrates what a two mode layer looks like

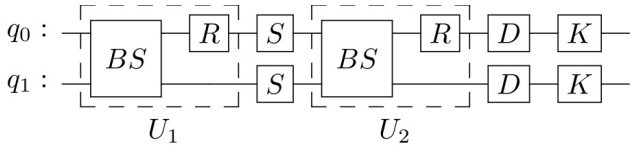


Figure 2: Layout of a two mode layer. U_1, U_2 are the interferometers made of beam splitter and rotation gates

Finally, in order to get an output after training, a measurement must be made. Here, we measure the expected value for the quadrature operator $\hat{x}$:

$$\langle \hat{x} \rangle = \langle \psi | \, \hat{x} \, | \psi \rangle \tag{4}$$

Once the measurement (which corresponds to the network's output) is made, we use a classical neural network to compute the loss function containing the physical constraints of the system and the weights are updated. Note that the weights which are trained are the parameters of the gate described in the previous section. This means that a QPINN can be trained with a very small number of parameters compared to a regular network.

Finally, the overall architecture of a quantum neural network can be summarized by the following figure (3):

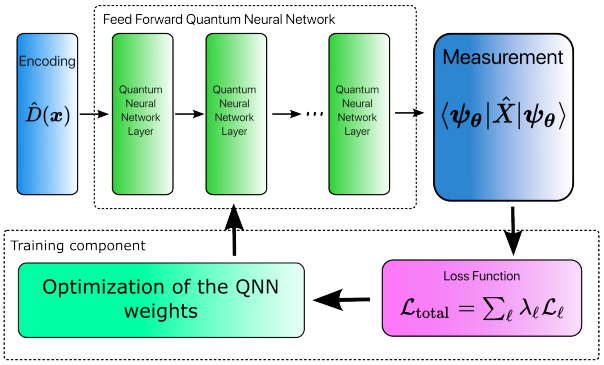


Figure 3: QPINN architecture, adapted from [2]; after an initial displacement are the several layers (defined as described above) which serve as a training step for the network. After measurement, the loss function is computed and the weights are updated using a classical neural network.

3.1 Training process

We use a standard hybrid loop. Pick collocation points in the domain (and on boundaries), run the circuit to get predictions at those points, use automatic differentiation (or parameter-shift rules) to compute derivatives with respect to inputs, assemble the residuals of the PDE(s), and update parameters by stochastic gradient descent (or Adam/L-BFGS). For the Poisson equation, we need second derivatives; nested AD handles this cleanly. For the Schrödinger–Poisson system we couple two residuals (wave and potential).

4 Research Design

We structure the experiments in three steps. Each adds one layer of complexity and checks a different aspect of the method.

Algorithm 1 QPINN training for the Poisson equation $\nabla^2 u(x) = s(x)$

Require: collocation points $\{x_i\}_{i=1}^N$, source $s(x)$, learning rate `lr`, epochs n

Initialize variational parameters $\theta \sim \mathcal{N}(0, \sigma^2)$

Define 4-layer QNN

for $t = 1$ **to** n **do**

  $L \leftarrow 0$

  for $i = 1$ **to** N **do**

    Prepare $|0\rangle$; encode x_i

    Apply all layers

    Measurement: estimate the solution to the equation $u_\theta(x)$

    Nested gradients: $u_x \leftarrow \partial_x u_\theta(x_i)$

    $u_{xx} \leftarrow \partial_x(u_x)$

    Residual: $r_i \leftarrow u_{xx} - s(x_i)$

    $L \leftarrow L + r_i^2$

  end for

  $L \leftarrow L/N +$ BC/IC penalties

  Optimizer (SGD, Adam...) update: $\theta \leftarrow \theta - \mathtt{lr}\, \nabla_\theta L$

end for

Return $\theta^\star$ and predictor $u_{\theta^\star}(x)$ =0

4.1 Problem A: 1D Poisson

Goal. Given a known source $g(x)$ on an interval and Dirichlet boundaries, recover the potential $\varphi(x)$:

$$\varphi''(x) = g(x), \qquad \varphi(0) = \varphi(1) = 0 \tag{5}$$

(or the analogous problem on $[0, \pi]$). We test two sources: $g(x) = x(x-1)$ and $g(x) = \sin(4x)$.

Circuit and loss. Inputs are positions x. The circuit outputs $\varphi_\theta(x)$. The loss is

$$L = \frac{1}{N} \sum_{i=1}^{N} \big(\varphi_\theta''(x_i) - g(x_i)\big)^2 + \lambda\big(\varphi_\theta(0)^2 + \varphi_\theta(1)^2\big). \tag{6}$$

We compute second derivatives via nested AD. Small depth (four layers) and two modes are usually enough.

The resulting output by the network is shown in the figure (4) below

4.2 Problem B: Heat equation

The heat equation describes how a field diffuses over time, for example temperature spreading along a rod. In our initial setup, we tried to follow the standard PINN recipe by taking the output $T(x, t)$ from the circuit, then computing its derivatives with respect to x and t

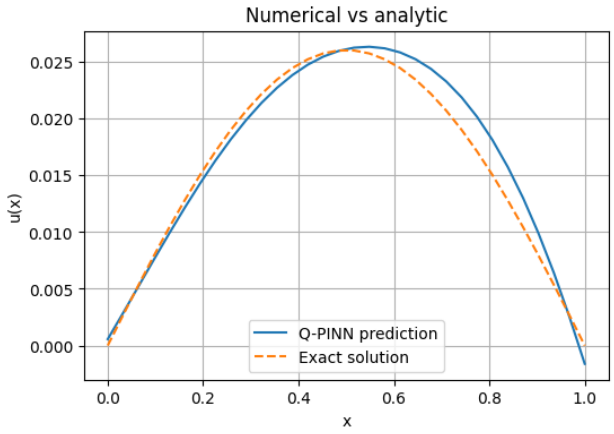


Figure 4: QPINN prediction for the 1D Poisson equation (quadratic case)

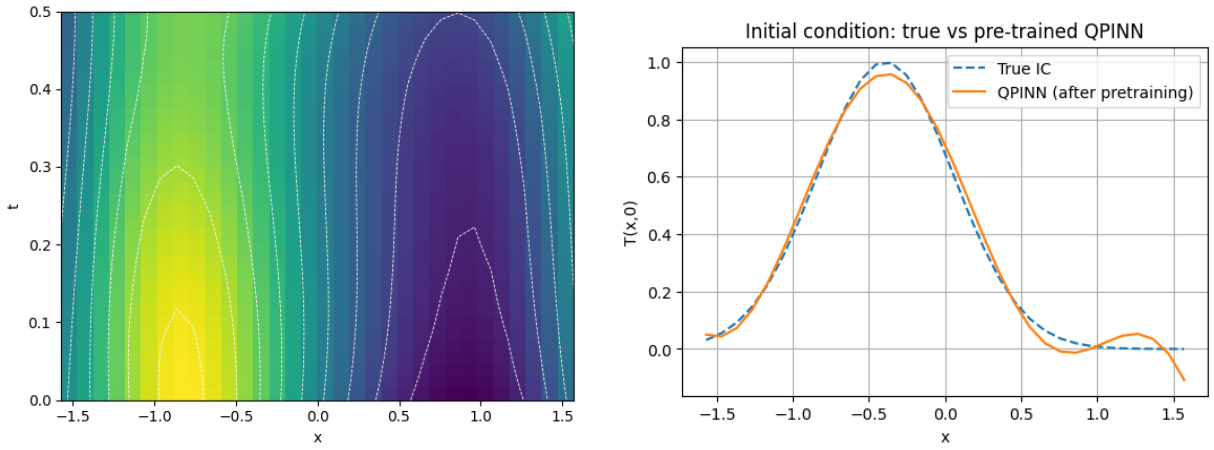
using nested gradients. However, this approach quickly ran into problems: nested differentiation through quantum circuits is both computationally expensive and numerically unstable, making training unreliable.

To address this, we introduced an additional qumode so that the circuit could output not only $T(x,t)$ but also its spatial derivative $T_x(x,t)$ directly. This allowed us to avoid nested gradients by defining a new loss function with two terms:

- A **consistency loss**, which enforces that the derivative of the first output matches the second output.
- A **PDE loss**, which uses the second output to enforce the heat equation relation between space and time derivatives.

This modified architecture makes the training much more stable and efficient. In practice, it provided better convergence and more accurate results compared to the naive nested-gradient approach.

The two figures below (5) show the result of initial condition training as well as the full domain solution.



(a) (b)

Figure 5: (a) QPINN prediction for the heat equation and (b) result of initial condition training

4.3 Problem C: Schrödinger–Poisson (FDM model)

The Schrödinger–Poisson system is the most ambitious case we considered, since it couples the wave-like dynamics of fuzzy dark matter to a self-consistent gravitational potential. Formally, the system is governed by

$$i\,\partial_t \Psi = -\tfrac{1}{2}\nabla^2\Psi + V\Psi, \qquad \nabla^2 V = |\Psi|^2 - 1,$$

with periodic boundary conditions. Because the wavefunction Ψ is complex, we must keep track of its real and imaginary parts in addition to the potential V.

As with the heat equation, our first approach was to compute all derivatives directly using nested gradients through the circuit. However, this quickly became unstable and computationally heavy, especially since we need both spatial and temporal derivatives of the real and imaginary parts, as well as the Laplacian of the potential. To overcome this, we increased the circuit to three qumodes, with each qumode dedicated to one output: the first for $\Re(\Psi)$, the second for $\Im(\Psi)$, and the third for V.

This change was necessary to fully capture the complex nature of the wavefunction. The full loss was therefore has several contributions:

- Schrödinger residual (real and imaginary parts),
- Poisson residual linking V and $|\Psi|^2$,
- Boundary penalties enforcing periodicity.

In practice, we attempted several setups (varying the number of layers, collocations points, initial training, optimizers...), all on the initial condition

$$\Psi(x,0) = \sqrt{1.0 + 0.6\sin\left(\tfrac{\pi x}{4}\right)}.$$

Both SGD and Adam optimizers were tested with different learning rates and cosine annealing schedules. Unfortunately, as of today, a fully stable solution was not achieved. However, this work highlighted clearly where the remaining difficulties lie: balancing the multiple residuals and maintaining numerical stability across the coupled fields.

5 Conclusion and Outlook

We built and tested QPINNs: physics-informed models where the function approximator is a small continuous-variable quantum circuit. The point was not to beat classical solvers on runtime but to check whether such circuits, trained against PDE residuals, can learn physically consistent solutions on problems that feature waves and self-consistent fields. On Poisson and heat, the QPINN performs like a standard PINN with similar capacity. On Schrödinger–Poisson, the circuit handles interference patterns naturally and the coupled loss steers it toward consistent potentials and densities for moderate times.

There are real limitations. Optimization remains delicate, loss balancing matters and it's important to keep in mind that everything here was done in simulation. On near-term hardware, noise and limited depth will constrain the circuits further. Still, the hybrid setup already has practical uses: compact surrogates that respect physics, quick "what if" evaluations once trained, and a natural platform for wave problems where basis choice is otherwise a headache.

In the near future, we will continue to work on this project in the hope of finding better processes to obtain efficient dark matter simulations. Assuming this is achieved, there is still a lot of very interesting work that could be conducted, like generalizing to three dimensions or running our project on actual quantum hardware. If nothing else, these experiments show that combining physics-informed training with quantum-inspired function classes is a viable path. Whether real quantum hardware will make it *better* depends on progress on depth, noise, and compilation—but the modeling recipe itself is sound and already useful in software.

Acknowledgments

I would like to particularly thank Emma Tolley and Ashutosh Mishra for accepting me in their lab and for all the precious guidance they provided. I also thank the Laidlaw Foundation for this amazing opportunity and experience that brought me a lot. Finally, this summer internship was as great as it was thanks to all the other Laidlaw scholars who were with me at EPFL and with whom I now share amazing memories !

References

[1] A. Mishra and E. Tolley, "SPINN: Advancing Cosmological Simulations of Fuzzy Dark Matter with Physics Informed Neural Networks," *The Astrophysical Journal*, 988:114, 2025. doi: 10.3847/1538-4357/ade43e

[2] G. Panichi, S. Corli and E. Prati, "Quantum physics informed neural networks for multi-variable partial differential equations," arXiv:2503.12244 [quant-ph], 2025.

[3] S. Markidis, "On physics-informed neural networks for quantum computers," *Front. Appl. Math. Stat.*, 2022, Article 1036711. doi: 10.3389/fams.2022.1036711

[4] N. Killoran, T. R. Bromley, J. M. Arrazola, et al., "Strawberry Fields: A software platform for photonic quantum computing," *Quantum*, 3:129, 2019.