Physics-Informed neural networks for the time-dependent Schrödinger equation

Afthash Puzhakkal,[1] Jonathan Marenkovic,[1] Alan Gardin,[1] and Giuseppe C. Tettamanzi[1]

[1]*Quantum and Nano Technology Group (QuANTeG), School of Chemical Engineering, The University of Adelaide, North Terrace Campus, Adelaide, 5000, South Australia, Australia*

Solving the time-dependent Schrödinger equation (TDSE) for high-dimensional systems can be computationally expensive using numerical methods such as Crank-Nicholson. Physics-informed Neural Networks (PINNs) offer a promising alternative by using deep learning techniques to find the solution by enforcing physical laws through loss functions. In this research project, my aim is to develop a PINN model to solve the TDSE. The study began with a single-electron system in a one-dimensional harmonic potential, then I approached a more complex problem, an electron in a moving quantum dot. These models were evaluated against known analytical or numerical solutions based on the probability density plot and wavefunction normalization plot. Later in the research, advanced training strategies such as causal training and normalization enforcement were explored, which improved the performance of the vanilla moving quantum dot model. Finally, I trained the model on a two-dimensional setup, demonstrating that the PINNs are capable of handling higher dimensions. In general, this research project explored a computationally efficient and scalable alternative to analytical and numerical methods for solving complex quantum problems.

CONTENTS

I. INTRODUCTION

The Schrödinger equation describes how the wavefunction of a quantum system evolves over time and space. Traditionally, the Schrödinger equation can be solved using analytical methods for simple systems and numerical methods for complex systems. Numerical methods such as Crank-Nicholson, while robust, become computationally expensive for higher-dimensional problems [1]. This creates the need for an efficient solver to solve higher-dimensional time-dependent quantum problems.

Physics-informed neural networks (PINNs), introduced by Raissi *et al.* [2], offer one such solver. They are deep learning models that learn to solve partial differential equations (PDEs) like the Schrödinger equation by including the physical laws directly into the model's loss function. The objective of this research is to understand whether physics-informed neural networks can solve the time-dependent Schrödinger equation accurately and efficiently.

I started the study by designing and creating a PINN architecture to solve the Schrödinger equation for a single electron trapped inside a one-dimensional harmonic potential. Later, I solved the TDSE for an electron inside a moving quantum dot. I verified the effectiveness of my models using the results from the Crank-Nicholson method. Later, to improve the effectiveness of the model, advanced training strategies such as causal training, where I ensure that the model does not violate causality during training [3], and normalization enforcement, where I explicitly force wavefunction normalization at selected times to stay at 1 during training, were implemented. Finally, I trained the dynamic part of the moving quantum dot system separately, both in 1-D and 2-D, to explore the movement of the quantum dot more closely.

II. RESEARCH PROPOSAL

a. Research gaps. The following needs to be checked. There are two gaps to be addressed in the PINN literature. The first is applicative. It seems well-known that PINN can lead to a computational advantage over numerical methods for high-dimensional problems. Yet, there are few applications. Hence, here we propose to apply PINN to a practical problem: the precise modelling of a single-electron source based on a nanowire quantum dot in silicon [4].

Second, there are several questions of fundamental interest that need to be solved. As an example, it has been shown how causality is necessary to ensure physical results [3]. Yet, there is no indication as to how one should choose the length of the time-segment over which causality needs to be enforced. The general consensus seems to be that answering such questions is problem specific. Thus, the problem we are choosing provides another example about how to apply these methods in practice. We hope that by developing more examples, researchers may be able to give general rule of thumbs in the future.

b. Proposed research. We consider a single-electron source constructed by a gate-defined quantum dot in a silicon nanowire, inspired by the work of Yamahata *et al.* [4], see figure 2 of the cited paper. The model is one-dimensional, and the electron wavefunction ψ obeys the time-dependent Schrödinger equation

$$i\hbar\frac{\partial\psi(x,t)}{\partial t} = -\frac{\partial^2\psi(x,t)}{\partial x^2} + V(x,t)\psi(x,t) \qquad (1)$$

where $m = 0.98m_e$ is the effective mass of electrons in silicon along the nanowire (m_e is the mass of an electron), and V the potential. The expression of the potential can be found in equation (S41) of the supplemental material of [4] (see file on Box).

This model is interesting for several reasons. First, modelling precisely a single-electron source at high frequency is of interest for researcher. Notably, Yamahata *et al.* [4] used a simple quantum dot approximation for the potential U. While this simplification captures the physics they wanted to discuss in their paper, it may also be limiting the predictive power. For instance, we hope that by using the real potential U, we may reproduce figure 5 of [4] so that it matches better with their experimental results.

A natural question is why did they not use the full potential in their paper? One reason may be that they were limited by the computational demand of this modelling. Indeed, a look at supplementary figure 6 shows that the potential in which the electron is trapped spans around 1.5 eV. Such a large energy scales may require too fine of a time-step to be solved by the Crank-Nicolson method. If such is the case, it would be interesting, and possible, to characterise the onset of when these problems become intractable on a computer. At this stage I can provide the following hints. (i) The energy scales determine the values of the onsite elements in a discretised Hamiltonian. If these energies have a large span, so do the onsite elements. It is know that numerical methods for finding eigenvalues, or inverting matrices, are sensitive to this. (ii) We are solving the time-dependent Schrödinger equation, which requires a time-step small enough to capture the non-adiabatic effects. This further complicates running a Crank-Nicolson solver, on top of the wide range of energy scales. If these computational hurdles are confirmed, we would have effectively found a much neglected class of problems for which PINNs could provide a computational advantage: problems with large energy scales.

Third, this problem could be made to include several electrons at the capture stage of the electron pump. Indeed, depending on the Fermi energy levels, initially two electrons could be trapped. When the potential barrier is raised, the electron-electron interaction are such that one elecron prefers to escape through the source. Such effects could be implemented through first-principles by electron-electron Coulomb interaction. Furthermore, it is of fundamental interest to address how the anti-symmetry of a fermionic wavefunction should be enforced in PINN.

Finally, this problem offers a novel research direction, that of Floquet PINNs. Floquet engineering, or Floquet driving, is widely used in quantum technologies, from the engineering of topological bandstructures in synthetic dimensions, to the control of quantum bit. Thus, it would be interesting to understand if a PINN can capture such time-dependent periodic perturbations over a full period. Note that this periodicity could also apply to other dimensions, such as spatial periodicity, hence opening applications to the modelling of bandstructures.

c. Proposed work. The first step will be to ensure that the initial condition is correctly found via Crank-Nicolson. To achieve this, return to the original quantum dot problem and solve for the ground state at $t = 0$. The ground state is simply the eigenvector corresponding to the smallest eigenvalue of the Hamiltonian. Then, compare the numerical solution to the analytically known answer, they should agree almost perfectly. Given the result is satisfactory, update the potential to the more complicated exact potential from ref. [4] and again obtain the ground state wave function as an initial condition.

Having obtained an initial condition for the complicated potential, use the Crank-Nicolson method to solve the time-dependent Schrödinger equation with the complicated potential from [4]. The aim is to compare the results in terms of accuracy and speed for the two methods available, PINN and Crank-Nicolson. Assuming that Crank-Nicolson can be made to work, begin exploring the computational advantage that may be gained by using PINN and how the performance of the different PINN models compare to each other and Crank-Nicolson.

III. LITERATURE REVIEW

Solving the Schrödinger equation has always been a central part of quantum mechanics. For most real-world systems, researchers have to rely on numerical methods such as the Crank-Nicholson method to solve the TDSE, which is accurate and reliable but becomes slow and expensive when solving time-dependent problems in higher dimensions.

The study by Gardin *et al.* [5] explored the behavior of an electron inside a moving quantum dot and solved the 2-D TDSE using the Crank-Nicholson method. Their work highlighted how the speed of a quantum dot affects the wavefunction's probability transitions. This study provides a strong physical background and reference data for applying PINNs to the moving quantum dot problem, making it relevant for this project.

In a paper by Raissi *et al.* [2], PINNs were introduced as a method to solve forward and inverse problems involving PDEs. Unlike traditional solvers, which follow a grid-based approach, PINNs follow a mesh-free approach and learn by minimizing the loss function, which involves the physical law described by the PDE. This makes PINNs a better choice where high-resolution or adaptive domains are required. Their work showed successful results in equations such as Burger's equation and the Navier-Stokes equation. This paper can act as a foundation for applying PINN to quantum mechanics.

Based on this, Shah *et al.* [6] applied PINNs to solve the TDSE for a 1-D harmonic oscillator. Their work showed that PINNs could predict the evolution of wavefunctions quite accurately, and they used known analytical solutions as a reference. However, they also found that performance dropped for longer time ranges and higher-energy states. To overcome this, they mentioned a method called causal training, which improves the overall performance of the model by training in a time-ordered manner. A further elaborative study about causality can be found in a paper by Wang *et al.* [3]. In that study, they proved that respecting the natural flow of time during training will help the model to learn more realistically.

Although PINNs have their strengths, they also have some limitations. Grossmann *et al.* [1] performed a detailed comparison between PINN and finite element method (FEM). They found that while PINNs are flexible, FEM is still more accurate and stable in many situations. This is especially true for large-scale problems or systems with stiff equations. Their work is a clear reminder that PINNs need improvements before they replace traditional solvers.

In summary, the literature shows that PINNs are a powerful new approach for solving quantum problems. They offer flexibility and grid-free modeling but still face accuracy and stability issues over longer time evolution. With additional techniques like causal training and normalization enforcement, this issue can be solved to an extent, and therefore PINNs can be considered as an alternative or complement to traditional solvers.

IV. METHODOLOGY

A. Physical modeling

The Schrödinger equation for a 1-D system, which describes how the wavefunction $\psi(x,t)$ evolves over time, can be expressed as:

$$i\hbar\frac{\partial\psi(x,t)}{\partial t} = -\frac{\hbar^2}{2m}\frac{\partial^2\psi(x,t)}{\partial x^2} + V(x,t)\psi(x,t) \tag{2}$$

Here, $\hbar$ is the reduced Plank's constant, m is the particle mass, and $V(x,t)$ is the potential energy function.

1. *One-dimensional static quantum dot*

My study began by recreating the baseline model from the paper by Shah *et al.* [6], where they solve the Schrödinger equation for an electron trapped in a 1-D harmonic potential. In this paper, they adopted Hartree atomic units ($\hbar = m_e = 1$), so energies are measured in Hartree and lengths in Bohr radii. The initial condition is given as a superposition of the ground and the first excited state.

The Schrödinger equation for this system will then be:

$$i\hbar\frac{\partial\psi(x,t)}{\partial t} = -\frac{\partial^2\psi(x,t)}{\partial x^2} + V(x)\psi(x,t) \tag{3}$$

where,

$$V(x) = \frac{1}{2}m\omega^2x^2 = \frac{1}{2}\omega^2x^2 \tag{4}$$

where ω determines the curvature of the potential. This corresponds to what we refer to as the baseline model in what follows.

2. *One-dimensional moving quantum dot*

Later, I extend my problem to a system where an electron is in a time-dependent moving quantum dot. From this problem onwards, I will be using nanometer (nm), picosecond (ps), and milli electron volt (meV) units. The new system's potential energy is:

$$V(x,t) = \frac{1}{2}m\omega^2(x - x_{qd}(t))^2 \tag{5}$$

In this context, $x_{qd}(t)$ denotes the position of the moving quantum dot at time t, as follows:

$$x_{qd} = \begin{cases} x_0, & t \le t_0 \\ x_0 + v_{qd}(t - t_1), & t_0 < t \le t_1 \\ x_1, & t \ge t_2 \end{cases} \tag{6}$$

where,

$$t_2 = t_1 + \frac{x_1 - x_0}{v_{qd}} \tag{7}$$

and,

- x_0: Initial position of quantum dot
- x_1: Final position of quantum dot
- t_0: Start time of the simulation
- t_1: Start time of quantum dot movement
- t_2: Stop time of quantum dot movement
- v_{qd}: Velocity of the quantum dot movement

This dynamic scenario is inspired by Gardin *et al.* [5], who explored non-adiabatic transitions in moving quantum dots.

3. One-dimensional dynamic-window

Later, I trained a new model to get a closer look at the dynamic part of the moving quantum dot system, which is the window between the time intervals t_1 and t_2 where the quantum dot actually moves in space. I will refer to this model as the dynamic-window model.

4. Two-dimensional dynamic-window

Finally, I created a 2-D version of the dynamic-window model, designed in a way such that the quantum dot moves through the (x, y) plane diagonally. The potential energy of the new system will be:

$$V(x,y,t) = \frac{1}{2}m\omega^2(x-x_0)^2 + \frac{1}{2}m\omega^2(y-y_0)^2 \tag{8}$$

where (x_0, y_0) is the position of the quantum dot in the (x, y) plane.

B. PINN Model Architecture

The PINN model takes position x and time t as inputs and gives the real and imaginary parts of the wavefunction ψ as output. I used a fully connected feedforward neural network with 6 hidden layers with 512 neurons each with an activation function tanh for the baseline PINN model and 12 hidden layers with 512 neurons each for the moving quantum dot model with the tanh activation function for the first layer and the SiLU activation function for the rest of the layers. The PINN learns by minimizing a loss function that enforces the governing PDE, initial condition, and boundary conditions.

C. Loss Function

As mentioned in earlier sections, a PINN model solves a problem by learning the physical laws of the system represented through a loss function. The loss function includes not only the PDE itself but also the initial and boundary conditions. The model will then be trained to minimize this loss function, which will result in learning the PDE and other conditions.

The total loss L_{total} can be defined as:

$$L_{total} = \lambda_{pde}L_{pde} + \lambda_{ic}L_{ic} + \lambda_{bc}L_{bc} \tag{9}$$

where,

- L_{pde}: Physics loss (PDE loss), handle the overall learning of the system's PDE
- L_{ic}: Initial condition loss, handle learning the wavefunction at $t = 0$
- L_{bc}: Boundary condition loss, handle learning the wavefunction at the spatial boundaries

Here $\lambda_{pde}, \lambda_{ic}, \lambda_{bc}$ are the regularization parameters or weights of the respective individual losses, through which we can control how much importance the model should give to learning each loss during training.

The residual of the Schrodinger equation, R_{pde}, can be written as:

$$R_{pde} = i\hbar\frac{\partial\psi(x,t)}{\partial t} + \frac{\hbar^2}{2m}\frac{\partial^2\psi(x,t)}{\partial x^2} - V(x,t)\psi(x,t) \tag{10}$$

and during training, model will learn the solution by minimizing this residual.
Since the wavefunction $\psi(x, t)$ is a complex number and PINNs can only handle real-valued variables, we need to rewrite it as:

$$\psi(x,t) = u + iv \tag{11}$$

we will now substitute this in equation 10 and separate the real and imaginary part of the residual. Which will give us:

$$R_{pde}^{real} = -\hbar\frac{\partial v}{\partial t} + \frac{\hbar^2}{2m}\frac{\partial^2 u}{\partial x^2} - V(x,t)u \tag{12}$$

and,

$$R_{pde}^{img} = i(\hbar\frac{\partial u}{\partial t} + \frac{\hbar^2}{2m}\frac{\partial^2 v}{\partial x^2} - V(x,t)v) \tag{13}$$

and using these values, we can define L_{pde} as:

$$L_{pde} = \frac{1}{N_{colloc}}\sum^{N_{colloc}} (R_{pde}^{real})^2 + (R_{pde}^{img})^2 \tag{14}$$

where N_{colloc} is the number of collocation points, or the number of samples used to calculate physics loss.

Similarly, we can enforce an analytical initial condition to the model's predicted u, v when $t = 0$. That will be the initial condition loss, or L_{ic}. Likewise, for the boundary condition, if we are using Dirichlet boundary conditions, both u and v should be 0 at spatial boundaries, and we can ensure this using boundary condition loss, or L_{bc}.

D. Data Sampling and Training

For the baseline model, I used collocation points $(x, t) \in [x_{min}, x_{max}] \times [t_{min}, t_{max}]$ following random uniform sampling, and for the rest of the models, the collocation points follow a random normal distribution, centered around the position of the quantum dot at a given time. Initial condition points are selected as uniform x points when $t = 0$, while boundary points are fixed at spatial domain edges.

The networks were trained for 250,000 epochs using the Adam optimizer with a learning rate $\gamma = 0.001$ that undergoes an exponential decay. I used the PyTorch framework in Python for building and training the models. Computations were performed on an NVIDIA A100 GPU.

E. Training Strategies

Everything up to now is essentially what we call the vanilla model. Now we will discuss the two additional training strategies extending the vanilla moving quantum dot model.

1. *Causal Training*

Causal training makes sure that the model's learning happens in a time-ordered manner. In other words, physics loss at later segments should depend on the same from earlier segments. Training the model this way will satisfy the principle of causality, which is inherent to quantum systems. To ensure causality, the physics loss is modified following Wang *et al.* [3] as:

$$L_{pde} = \frac{1}{N_{seg}} \sum_{i=1}^{N_{seg}} \left(\sum_{k=1}^{i-1} L_k\right) + L_i \tag{15}$$

where,

- N_{seg} is the number of segments used
- L_i and L_k are L_{pde} given by eq. (14) for segment i and k respectively

For the rest of this report, the model in which causality is enforced will be referred to as the causal model.

2. *Normalization Enforcement*

When solving Schrödinger equation, normalizing the wavefunction during the simulation ensures that the wavefunction follows a valid probability distribution. In other words, the probability of finding a particle anywhere in the spatial domain should stay at 1 all the time. Mathematically, this can be represented as,

$$\int |\psi(x,t)|^2 dx = 1, \tag{16}$$

But when training a model using PINNs, we don't explicitly normalize the wavefunction; instead, the final wavefunction is expected to stay normalized as a result of the model learning the system correctly.

As part of improving my vanilla quantum dot model, I decided to manually enforce the wavefunction to stay normalized during training with the help of an additional term to the loss function. During training, I used the Monte Carlo method to find an approximation of the integration represented in equation 16. The Monte Carlo approximation at time t can be expressed as:

$$R_{norm} = (x_{max} - x_{min}) \frac{\sum_{i=0}^{N_{mc}} \psi(x_i, t)}{N_{mc}} \tag{17}$$

where R_{norm} will be the normalization residual and N_{mc} is the number of wavefunction samples we used to find the approximation. Building on top of this, the normalization loss, or L_n, can be defined as:

$$L_n = \frac{1}{N_{norm}} \sum^{N_{norm}} (R_{norm} - 1)^2 \tag{18}$$

so that minimizing L_n will bring the value of R_{norm} closer to 1. Here, N_{norm} is the number of equidistant time points we selected to find the normalization loss. If we included L_n in the total loss (equation 9), the new total loss will be,

$$L_{total} = \lambda_{pde} L_{pde} + \lambda_{ic} L_{ic} + \lambda_{bc} L_{bc} + \lambda_n L_n \tag{19}$$

where λ_n is the weight of normalization loss.

For the rest of this report, the model in which wavefunction normalization is enforced will be referred to as the normalization model or the norm model.

F. Model Evaluation

I used the results from an analytical equation to evaluate the baseline model and results from the Crank-Nicholson method to evaluate the moving quantum dot models. The performance was evaluated using a comparison of the probability density plots and the wavefunction normalization plots.

The first metric used for evaluation is the maximum absolute error between the probability density predicted ($\Delta\psi^2(max)$) by the PINN model and the result from the analytical/numerical method, which can be denoted by:

$$\Delta\psi^2(max) = max\left(\left| |\psi|^2_{predicted} - |\psi|^2_{actual} \right|\right) \tag{20}$$

where $|\psi|^2_{predicted}$ and $|\psi|^2_{real}$ are the probability densities of the predicted result and the analytical/numerical result, respectively.

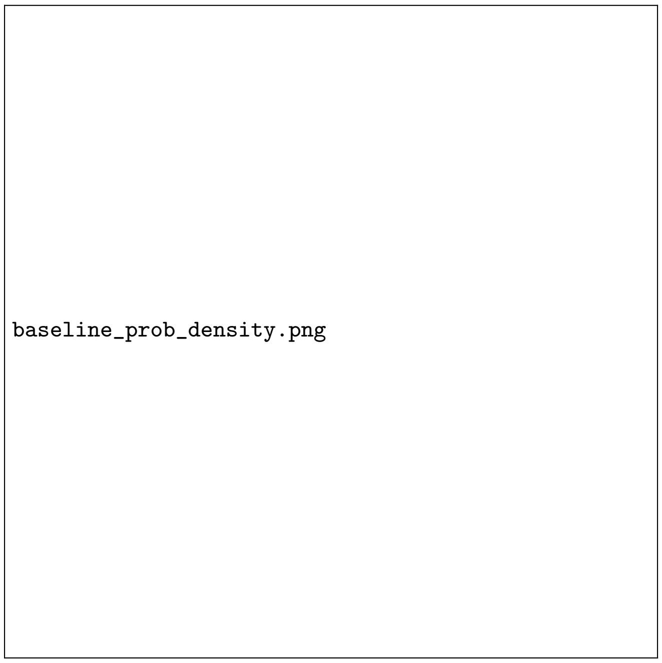

FIG. 1. Probability Density $|\psi(x,t)|^2$ of the baseline model. Here, I reproduced the results from the paper by Shah *et al.* [6], and this plot can be verified by comparing figure 1 in that paper.

$\Delta\psi^2$	**Wavefunction Normalization**		**Training Time**
(max)	**Mean**	**Standard deviation**	**(minutes)**
0.00940	0.99966	0.00020	71

TABLE I. Performance metrics of the Baseline model

The other two metrics are the mean and standard deviation of the wavefunction normalization. This will help to determine how much the predicted solution deviates from the physically accurate solution.

V. RESULTS

A. 1-D static quantum dot

I started this project by developing a PINN model for a system where an electron is trapped in a 1-D harmonic oscillator. This baseline model was able to predict the wavefunction evolution correctly. The predicted probability density $|\psi(x,t)|^2$ (figure 1) closely matched the analytical results.

B. 1-D moving quantum dot

As a next step in the project, I trained the PINN model to solve a system where an electron is inside a time-dependent moving quantum dot. The probability density of this system's Crank-Nicholson solution is represented in figure 2.

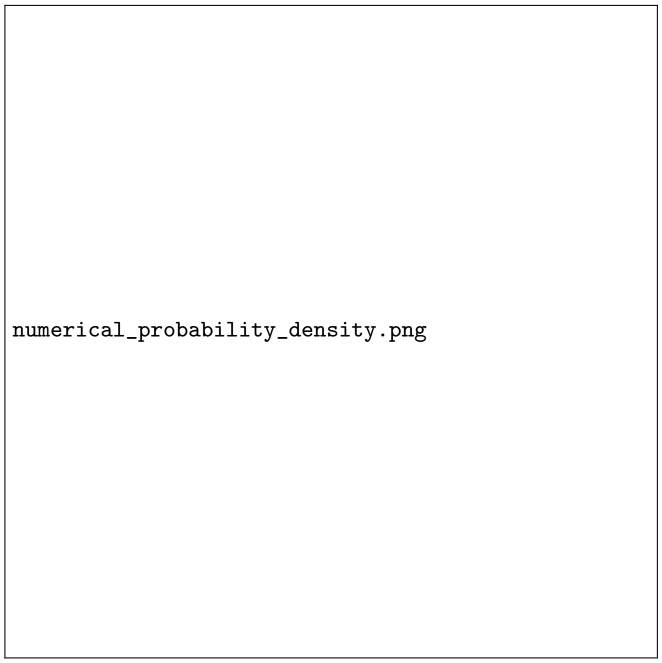

FIG. 2. Crank-Nicholson result of an electron inside a moving quantum dot system's probability density. Here, the quantum dot is moved from $x_0 = 0$ nm to $x_1 = 75$ nm with a velocity $v_{qd} = 15$ nm/ps. This resulted in the electron inside the quantum dot undergoing non-adiabatic oscillation after the quantum dot stopped moving. In the figure, the red dotted line represents the position of the quantum dot at a given time. This result will be used to evaluate the performance of PINN models.

1. *Vanilla Model*

The vanilla PINN model was able to capture the overall motion of the wavefunction. However, as time progressed, the accuracy decreased. From figure 3, which is the probability density plot, you can see that the shape of quantum dot's movement and the electron's oscillations after the quantum dot stopped moving are correctly predicted. But towards the end of the simulation, a clear loss of accuracy is observed.

This can be clearly understood from the absolute error plot between the vanilla model and Crank-Nicholson results, as depicted in figure 4, and the wavefunction normalization plot in figure 7, where the vanilla model's probability reaches as low as 0.86, with a mean of 0.90468, as shown in table II.

2. *Effect of Causality*

To improve the performance of the vanilla model, I experimented with a causal training strategy. Through `qdot_probability_density.png`

FIG. 3. Probability density $|\psi(x,t)|^2$ of the vanilla moving quantum dot model. Here, you can see that, while the quantum dot trajectory and electron's oscillation are correctly predicted, there is a clear loss of accuracy, which is visible towards the end of the simulation. This error accumulation is clearly visible in figure 4 where I plot the absolute error of probability density between the PINN prediction and Crank-Nicholson result.

`qdot_probability_density_error.png`

FIG. 4. Absolute error of $|\psi(x,t)|^2$ between the PINN prediction and Crank-Nicholson result. Here you can see that the error increases as the time progresses. This error accumulation is due to the PINN's inability to handle longer time evolution.

Model	$\Delta\psi^2$ **(max)**	**Wavefunction Normalization**		**Training Time (minutes)**
		Mean	**Standard deviation**	
Vanilla	0.03409	0.90468	0.04191	394
Causal	0.00607	0.98813	0.00447	381
Norm	0.00982	1.00111	0.00023	492

TABLE II. Performance metrics of the moving quantum dot models. Here, you can see that both the causal and normalization enforcements clearly improved the vanilla model. The causal model has improved the most on the maximum absolute error, while the norm model improved the wavefunction normalization metrics. One thing to note here is that causal training does not increase the training time, but the normalization model does, and that is because for the normalization model, we are using an additional loss function.

`causal_probability_density.png`

FIG. 5. Probability density $|\psi(x,t)|^2$ of the causal model. Here, the error accumulation is decreased drastically when compared to the same plot of the vanilla model (figure 3. This accuracy improvement is more obvious when you refer the wavefunction normalization plot (figure 7) where you can see how much the vanilla model improves when done causal training.

training the model in a time-ordered manner, it was possible to satisfy the principle of causality, which resulted in the model maintaining it's accuracy over time.

The improvement is clearly visible in table II or when we compare the probability density plot of the causal model (figure 5) with that of the vanilla model (figure 3).

3. *Effect of Normalization Enforcement*

Another training strategy I used to improve the vanilla model is enforcing normalization during training. As you

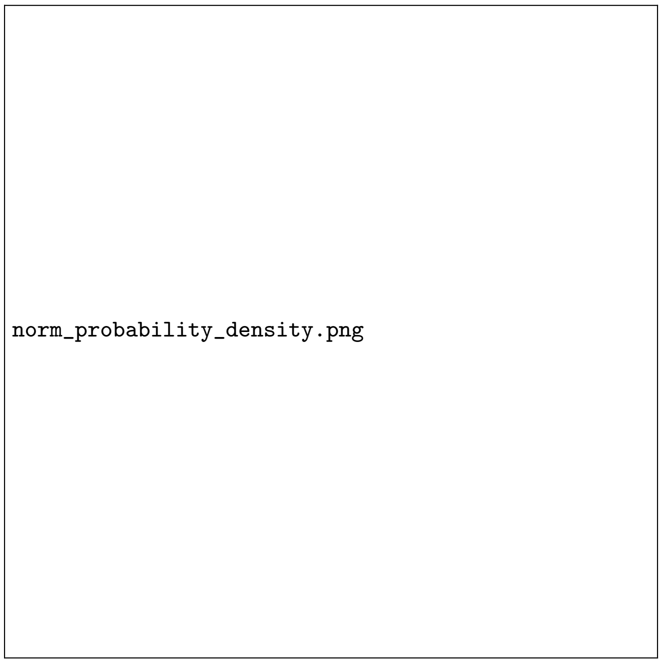

FIG. 6. Probability density $|\psi(x,t)|^2$ of the normalization model.

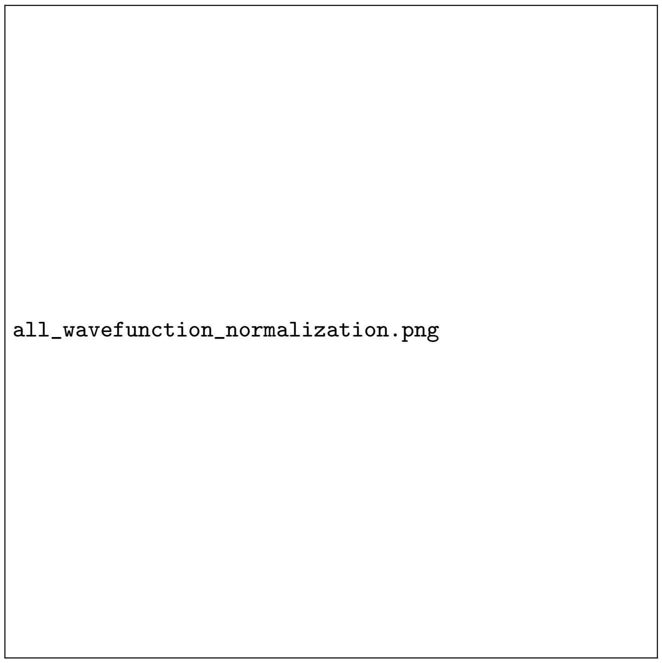

FIG. 7. Wavefunction normalization plots of the moving quantum dot models. You can see that the vanilla model deviates the most from 1 as time progresses. This issue is clearly improved by the causal model, which still shows a loss of accuracy over time; the rate of loss has been reduced by a lot when compared to the vanilla model. The norm model, where I enforce normalization, shows the effectiveness of the training strategy by staying closer to 1 all the time, without showing any loss of accuracy.

can see earlier in figure 3, the vanilla model produces non-physical results towards the end of the simulation because the wavefunction normalization deviates too much from 1.

After enforcing normalization, the normalization model consistently kept the total probability close to 1 throughout the simulation, as shown in figure 7. This resulted in the probability density plot improving (figure 6) when compared to that of the vanilla model (figure 3).

C. 1-D dynamic-window

From the results of the moving quantum dot models, I found out that the PINN struggles to learn the non-adiabatic oscillations part correctly for the vanilla model (figure 3). Even though the training strategies clearly improved the vanilla model, a question still arises: why did the vanilla model fail to learn the system properly? One of the reasons could be the fact that I am training the model for a longer time range, while PINNs have an issue with longer time evolutions [6].

A potential way to overcome this issue, other than the training strategies of course, is to train for a shorter time range. In this section, I will be showing the improvements of the model when trained only for the dynamic movement of quantum dot.

To compare it's performance, if we look at the probability density plot (figure 8), we can see that unlike the vanilla model (figure 3), there are no visible loss of accuracy. This is achieved with less training time compared to the vanilla model, as shown in table III.

Model	**$\Delta\psi^2$ (max)**	**Wavefunction Normalization**		**Training Time (minutes)**
		Mean	**Standard deviation**	
Vanilla*	0.01484	0.94422	0.01527	394
D-W	0.00064	0.99958	0.00084	110

TABLE III. Comparison between the vanilla model and the dynamic-window model (D-W).
Please note that here the vanilla model is only predicting the movement part of the quantum dot (t_1 to t_2), the time range in which the dynamic-window model is trained.

D. 2-D dynamic-window

After noticing an increase in model accuracy with a less complex model for the Dynamic-Window model, I decided to train a model of the same system in two dimensions. Solving the TDSE for a 2-D system will give a clear idea about how both PINN and Crank-Nicholson handle the complexity of higher dimensions.

The PINN model not only predicted the wavefunction for the 2-D Dynamic-Window system (figure 9), but also outperformed the Crank-Nicholson method both in training time and final output file size, as shown in table IV. The results show the robustness of PINN in handling td_probability_density.png

FIG. 8. Probability density $|\psi(x,t)|^2$ of the dynamic-window model.

2d_probability_density.png

FIG. 9. Probability Density $|\psi(x,y,t)|^2$ for the 2-D dynamic-window model

higher-dimensional problems.

E. Summary of results

- The baseline model successfully learned the 1-D harmonic oscillator system.

Solver	Training/simulation time (minutes)		Output size	
	1-D	2-D	1-D	2-D
PINN	110	385	5.1 MB	5.1 MB
Crank-Nicholson	0.066	443	62 MB	15 GB

TABLE IV. Training/simulation time for both the 1-D and 2-D dynamic-window models using the PINN and Crank-Nicholson method. Here, you can clearly see that PINN scales better to 2-D when compared to the Crank-Nicholson method, whose simulation time and output file size explode when scaled to 2-D.

- For the vanilla moving quantum dot model, loss of accuracy was visible towards the end of the simulations.

- Causal training helped the moving quantum dot model to improve its stability.

- Similarly, normalization enforcement reduced the loss of accuracy towards the end of the time domain.

- Focusing the training only on the movement part of the quantum dot made the training more efficient and results more accurate, as shown in the results of the dynamic-window model.

- The 2-D dynamic-window model proved the PINNs' capability to handle higher dimensions better than the Crank-Nicholson method.

VI. DISCUSSION

This research project set out to explore whether PINNs can accurately solve the TDSE, especially for systems that involve a time-dependent moving potential. Based on the results, the answers are generally positive.

Starting with the 1-D harmonic oscillator, PINN showed decent performance with respect to the known analytical solution. This confirmed that the model could learn basic quantum behavior correctly [6].

However, when I tried to implement PINN to the moving quantum dot model, I faced several challenges. Firstly, I had to double the model complexity, especially to capture the electron's non-adiabatic oscillation after the quantum dot stopped moving. In addition to this, the model showed a clear loss of accuracy over longer simulation times. This is an issue PINNs face when following standard training methods [1]. The vanilla model also failed to maintain proper normalization of the wavefunction, which is critical in quantum mechanics.

To address this, I implemented two different training strategies: causal training and normalization enforcement. Causal training helped guide the model to learn in a time-ordered manner, just like in real quantum systems [3]. This improved the stability of predictions and reduced error accumulations. Similarly, enforcing normalization during training ensured that the total probability remained constant throughout the training, and this helped the model to reduce the loss of accuracy, especially towards the end of the time domain.

Another method I tried to capture the best out of PINN is to train the model for only the part where the quantum dot moves, in other words, only the dynamic-movement part. Since the training time range is reduced, this resulted in the model improving its performance. While this approach limits generalization outside the selected time window, it is still practical for targeted quantum simulations. Finally, the same system was tested in two dimensions. PINN handled this complexity more effectively than the Crank-Nicholson method, highlighting its upper hand in handling higher-dimensional problems.

Overall, this project's results show that, while PINNs are not perfect, they can be powerful tools for solving TDSE when combined with good design choices, physics-based constraints, and training strategies. These results are important for applications in quantum computing, quantum control, and time-dependent simulations.

VII. CONCLUSION

In this research project, I used PINNs to solve the TDSE for both simple and complex quantum systems. The results showed that, while PINNs can easily learn solutions for simple problems, they might struggle with more complex problems.

By introducing training strategies such as causal training and normalization enforcement, the model was able to bring the performance of the complex systems closer to that of simple systems. This suggests that PINNs can be a practical and flexible tool for solving time-dependent quantum problems.

Future works could look at improving training strategies, creating a more hybrid model that combines both PINN and traditional numerical methods, and more generalized models that include variables such as frequency ω, quantum dot movement speed v_{qd}, and time-dependent potential energy $V(x,t)$ as inputs.

ACKNOWLEDGMENTS

This work was supported with super computing resources provided by the Phoenix HPC service at the University of Adelaide.

[1] T. G. Grossmann, U. J. Komorowska, J. Latz, and C.-B. Schönlieb, Can physics-informed neural networks beat the finite element method?, IMA Journal of Applied Mathematics **89**, 143 (2024).

[2] M. Raissi, P. Perdikaris, and G. E. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, Journal of Computational physics **378**, 686 (2019).

[3] S. Wang, S. Sankaran, and P. Perdikaris, Respecting causality is all you need for training physics-informed neural networks, arXiv preprint arXiv:2203.07404 (2022).

[4] G. Yamahata, S. Ryu, N. Johnson, H.-S. Sim, A. Fujiwara, and M. Kataoka, Picosecond coherent electron motion in a silicon single-electron source, Nature Nanotechnology **14**, 1019 (2019).

[5] A. Gardin, R. D. Monaghan, T. Whittaker, R. Rahman, and G. C. Tettamanzi, Nonadiabatic quantum control of valley states in silicon, Physical Review B **105**, 075406 (2022).

[6] K. Shah, P. Stiller, N. Hoffmann, and A. Cangi, Physics-informed neural networks as solvers for the time-dependent schrödinger equation, arXiv preprint arXiv:2210.12522 (2022).

Appendix A: Additional Plots

`baseline_probability_density_error.png`

FIG. 10. Absolute error of probability density between the baseline model prediction and analytical solution.

`baseline_wavefunction_normalization.png`

FIG. 11. Wavefunction normalization of the baseline model

`causal_probability_density_error.png`

FIG. 12. Absolute error of probability density between the causal model prediction and numerical solution.

`norm_probability_density_error.png`

FIG. 13. Absolute error of probability density between the norm model prediction and numerical solution.

td_probability_density_error.png

FIG. 14. Absolute error of probability density between the dynamic-window model prediction and numerical solution.

td_wavefunction_normalization.png

FIG. 15. Wavefunction normalization of the dynamic-window model

2d_wavefunction_normalization.png

FIG. 16. Wavefunction normalization of the 2-D dynamic-window model