A structure-preserving neural ODE framework for learning potentials in the time-dependent Schrödinger equation

Veronica Nikiforova and Bach Luu

July 2026

1 Introduction and problem description

The time-dependent Schrödinger equation (TDSE) governs the evolution of a quantum system's wave function ψ over time. The wave function encodes the system's state and, when squared, yields the probability density of locating a particle at a given position [1]. The TDSE underpins a wide range of applications, such as the simulation of ultrafast electron dynamics [2] and attosecond-scale quantum control at the nanoscale [3]. These advances have significantly influenced fields including quantum chemistry, nanotechnology, and quantum information science.

In one dimension, the TDSE is given

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

where $\psi(x,t)$ is the complex-valued wavefunction, $V(x,t)$ is the potential function, m is the particle mass, and $\hbar$ is the reduced Planck constant. We will set m and $\hbar$ to natural units ($m = 1, \hbar = 1$) to simplify this expression, and we also will first examine a time-invariant potential, $V(x)$:

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

In many physical settings, the potential $V(x)$ is unknown or partially observed, yet it plays a crucial role in determining the system's dynamics. We typically have access to indirect observations, such as the state of the system at different points in time. In this project, we aim to learn the potential $V(x)$ from limited information: a pair of wavefunctions $\psi_0 = \psi(x,t_0)$ and $\psi_1 = \psi(x,t_1)$ that represent the system at two different times.

This project investigates whether it is possible to infer the potential from such observations. This inverse problem is inherently challenging, as multiple potentials can lead to similar observed behavior, and small changes in the data can significantly affect the inferred solution. Despite these challenges, solving this problem could provide valuable insights into systems where direct measurement of the potential is not feasible, offering a new pathway to understanding complex quantum dynamics. We employ a machine learning framework to infer the potential $V(x)$ from limited observational data. While machine learning has been studied for decades, its impact has grown dramatically in recent years due to increased data availability and improvements in computational tools [4]. In particular, neural Ordinary Differential Equations (neural ODEs) offer a powerful archetype for modeling systems with continuous-time dynamics [5]. By embedding a neural network representation of the potential $V(x)$ within an ODE framework, we can learn the structure of the system from sparse measurements while ensuring consistency with the underlying physical laws. This continuous formulation also enables greater flexibility and generalization in modeling complex dynamical behavior.

2 The classic neural ODE framework and its limitations

In this section, we first present the neural ODE [5] architecture, viewing a deep network as a continuous-time flow driven by a learnable vector field. Next, we outline how gradients are computed via the adjoint sensitivity method, which integrates another ODE backward in time and thereby sidesteps the need to store the entire forward trajectory. We then turn to our specific context, the time-dependent Schrödinger equation, and identify why we cannot apply this framework directly to solve the problem. These observations motivate the Crank–Nicolson adjoint scheme introduced in the following section.

2.1 Neural ODEs and the adjoint method

The neural ODE [5] framework is a deep learning structure introduced to model continuous-time dynamics using neural networks. Instead of stacking discrete layers like in a residual network (ResNet) or other traditional neural network structures, a neural ODE defines the hidden state $h(t)$ of a system as the solution to an ordinary differential equation:

$$\frac{dh(t)}{dt} = f_\theta(h(t), t), \tag{3}$$

where f_θ is a neural network parametrized by θ and $h(t_0)$ is the initial condition. The output $h(t_1)$ is computed by integrating this ODE numerically from t_0 to t_1. This approach enables the modeling of smooth, continuous time dynamics and allows for efficient backpropagation via the adjoint sensitivity method. In traditional backpropagation through an ODE solver, the entire trajectory of the hidden state over time would need to be stored, a potentially memory-intensive task. The adjoint sensitivity method avoids this by reframing the computation of gradients as the solution to a second, augmented ODE that runs backward in time.

Suppose we define a scalar loss function $L(h(t))$ where $h(t)$ is the solution of Equation (3). The adjoint method defines an additional variable, the adjoint state $a(t) = \frac{\partial L}{\partial h(t)}$, which satisfies its own ODE:

$$\frac{da(t)}{dt} = -\frac{\partial f_\theta}{\partial h} a(t)^T. \tag{4}$$

The derivation of this adjoint variable can be found in Appendix B of the paper on neural ordinary differential equations [5]. This adjoint variable allows the gradient $\frac{dL}{d\theta}$ to be computed by solving another backward ODE that accumulates sensitivities with respect to the parameters θ:

$$\frac{dL}{d\theta} = -\int_{t_1}^{t_0} \frac{\partial f_\theta}{\partial \theta} a(t)^T dt. \tag{5}$$

Thus, we get a system of three ODEs, as follows, with C representing $\frac{dL}{d\theta}$:

$$\begin{aligned}
\frac{dh}{dt} &= f_\theta(h(t), t), \text{ initial condition}\, h_0 = h(t_0), \text{ solved from } t_0 \text{ to}\, t_1 \\
\frac{da}{dt} &= -\frac{\partial f_\theta}{\partial h(t)} a(t)^T, \text{ terminal condition}\, a(t_1) = \frac{\partial L}{\partial h(t_1)}, \text{ solved from } t_1 \text{ to}\, t_0. \\
\frac{dC}{dt} &= -\frac{\partial f_\theta}{\partial \theta} a(t)^T, \text{ terminal condition, } C(t_1) = 0, \text{ solved from } t_1 \text{ to}\, t_0
\end{aligned} \tag{6}$$

Each of these is solved, as described above, in order to calculate our final state $h(t_1)$ and the accumulated gradient of the loss with respect to the neural network parameters θ in place of performing standard backpropagation.

In our setting, the adjoint method is adapted to work with the Crank–Nicolson time-stepping scheme used to solve the TDSE, where ψ plays the role of the hidden state, and the dynamics are implicitly controlled by the neural network through the potential $V_\theta(x)$.

2.2 Learning from structure-preserving dynamics

Neural ODEs [5] cast deep networks as continuous-time flows whose trajectories are obtained by any numerical solver. Such as in the reference implementation at Github Repository, these solvers are general-purpose schemes such as Euler, classic and adaptive Runge-Kutta, Adams, etc. that optimize local truncation error but do not enforce geometric or analytic invariants. As a consequence, a model trained with this standard approach can fit observed data while silently violating constraints such as conservation, symmetries, or energy dissipation. This problem is particularly acute for quantum systems governed by TDSE, whose evolution is exactly unitary and time-reversible.

The TDSE possesses fundamental mathematical properties that must be preserved in any neural ODE implementation to ensure physical consistency. One of the most crucial of these properties is unitarity, which guarantees probability conservation and reversible quantum dynamics. The probability of finding a particle in region $[a, b]$ is:

$$P(x \in [a, b]) = \int_a^b |\psi(x, t)|^2 \, dx \tag{7}$$

For this probabilistic interpretation to be meaningful, the total probability must equal one at all times.

Equally important to probability conservation is the time-reversibility of unitary evolution. This means any forward evolution can be undone exactly: propagating for a time t and then for $-t$ returns the wave-function to its initial state. Below are the proofs that the TDSE possesses both of these properties.

Theorem 1 (Norm conservation of the TDSE)**.** *Suppose that ψ satisfies*

$$i\,\frac{\partial \psi}{\partial t} = H\psi, \quad H = -\frac{1}{2}\,\frac{\partial^2}{\partial x^2} + V(x), \tag{8}$$

then the L^2 norm of ψ is conserved for all t, which means

$$\int_{-\infty}^{\infty} |\psi(x,t)|^2\,dx \;=\; \int_{-\infty}^{\infty} |\psi(x,0)|^2\,dx. \tag{9}$$

Proof. Differentiate the norm:

$$\frac{d}{dt}\int_{-\infty}^{\infty} |\psi|^2\,dx \;=\; \int_{-\infty}^{\infty} \Big(\frac{\partial \psi^\dagger}{\partial t}\,\psi + \psi^\dagger\,\frac{\partial \psi}{\partial t}\Big)dx. \tag{10}$$

From the Schrödinger equation, we have

$$\frac{\partial \psi}{\partial t} = -iH\psi, \quad \frac{\partial \psi^\dagger}{\partial t} = iH\psi^\dagger, \tag{11}$$

since hermiticity gives $H^\dagger = H$. Substitute these expressions to obtain

$$\frac{d}{dt}\int_{-\infty}^{\infty} |\psi|^2\,dx \;=\; \int_{-\infty}^{\infty} i(\psi^\dagger H\psi - \psi^\dagger H\psi)\,dx \;=\; 0, \tag{12}$$

because the two integrands cancel exactly. Hence, the norm is preserved for all time. □

Theorem 2 (Time-reversibility of the TDSE)**.** *If we have*

$$i\,\frac{\partial \psi}{\partial t} = H\psi, \qquad H = -\frac{1}{2}\,\frac{\partial^2}{\partial x^2} + V(x), \qquad \psi(0) = \psi_0, \tag{13}$$

then with $U(t) = e^{-iHt}$ we have

$$\psi(t) = U(t)\psi_0, \qquad U(-t) = U(t)^\dagger = U(t)^{-1}, \tag{14}$$

and hence

$$\psi_0 = U(-t)\,\psi(t). \tag{15}$$

Proof. Hermiticity gives $H^\dagger = H$. Therefore,

$$U(t)^\dagger = \big(e^{-iHt}\big)^\dagger = e^{\,iH^\dagger t} = e^{\,iHt} = U(-t) \tag{16}$$

Hence, $U(t)^\dagger U(t) = U(-t)U(t) = I$ and $U(t)^\dagger = U(t)^{-1}$. Since $\psi(t) = U(t)\psi_0$, multiplying by $U(-t)$ yields $\psi_0 = U(-t)\psi(t)$. □

3 A structure-preserving neural ODE framework

We approach the problem of recovering $V(x)$ by using a neural network to learn from data generated through numerical simulations. Our core idea is to use this neural network to parameterize the unknown potential, and then evaluate its quality by evolving the system forward under the TDSE. We discretize the spatial domain, reducing the TDSE from a PDE to an ODE in time. This enables us to leverage the neural ODE framework, where the dynamics are governed by a known physical law: the TDSE. In this case, the neural network appears only through its role in defining the potential $V_\theta(x)$.

To generate training data, we use the Crank–Nicolson method, a structure-preserving numerical scheme that ensures unitarity and norm conservation during time evolution [6]. Each training pair consists of an initial wavefunction ψ_0 and its evolved counterpart ψ_1 under a known potential. During training, the model proposes a candidate potential $V_\theta(x)$, integrates the TDSE forward from ψ_0, and compares the predicted wavefunction $\hat{\psi}_1$ to the true ψ_1.

This setup allows the model to learn physically consistent potentials by minimizing a loss based on wavefunction mismatch. To train efficiently, we use the adjoint method to compute gradients of the loss with respect to the potential parameters.

The following sections describe in detail how the neural ODE framework is adapted to this setting and how the Crank–Nicolson method is used to preserve key physical properties throughout the training process.

3.1 Properties of the adjoint dynamics in the Schrödinger equation

In our project, we adapt this neural ODE framework to the TDSE, which becomes an ODE in time for the wavefunction $\psi(x,t)$ after discretizing the spacial domain. Specifically, by replacing the continuous Laplacian operator ($\frac{\partial^2\psi}{\partial x^2}$) with a discrete second-derivative matrix (which uses finite difference approximations), the wavefunction $\psi(x,t)$ is represented as a vector $\psi(t)$ of complex amplitudes over spacial grid points. The TDSE then takes the form

$$\frac{d\psi}{dt} = f_\theta(\psi, t) = -i\Big(-\frac{1}{2}\frac{\partial^2}{\partial x^2} + V_\theta(x)\Big)\psi, \tag{17}$$

where $V_\theta(x)$ is the diagonal matrix containing the potential values at each grid point. Rather than learning the full dynamics directly with a neural network f_θ, we use a neural network to parameterize only the potential $V_\theta(x)$. This potential is plugged into the TDSE to define the right-hand side of the ODE that governs the evolution of ψ. To evaluate the accuracy of a candidate potential $V_\theta(x)$, we integrate the TDSE (using the Crank-Nicolson method) from ψ_0 to produce a predicted wavefunction $\hat{\psi_1}$. The loss function is based on the discrepancy between $\hat{\psi_1}$ and ψ_1, and we use the adjoint method to compute gradients of this loss with respect to θ, enabling efficient training.

The extended dynamics for the adjoint sensitivity method are derived from the standard formulations of the general neural ODE framework, but adjusted to account for the fact that the only reliance our function f_θ has on θ is through the potential, $V_\theta(x)$. They are defined as follows, with $H = -\frac{1}{2}\frac{\partial^2}{\partial x^2} + V_\theta(x)$:

$$\begin{aligned}
a(t) &= \frac{\partial L}{\partial \psi}, \\
\frac{da(t)}{dt} &= -\frac{\partial f_\theta}{\partial \psi}a(t)^T = -Ha(t)^T, \\
\frac{dL}{d\theta} &= -\int_{t_1}^{t_0} \frac{\partial f_\theta}{\partial \theta}a(t)^T dt = -\int_{t_1}^{t_0} (-i)a(t)^T \frac{dV_\theta}{d\theta}\psi(t)dt.
\end{aligned} \tag{18}$$

Thus, our system of ODEs that are solved to attain the final state $\hat{\psi_1}$ and calculate the gradient of the loss is:

$$\begin{aligned}
\frac{d\psi}{dt} &= f_\theta(\psi,t), \text{ initial condition}\, \psi_0 = \psi(t_0), \text{ solved from } t_0 \text{ to}\, t_1 \\
\frac{da}{dt} &= -Ha(t)^T, \text{ terminal condition}\, a(t_1) = \frac{\partial L}{\partial \psi_1}, \text{ solved from } t_1 \text{ to}\, t_0 \\
\frac{dC}{dt} &= ia(t)^T\frac{dV_\theta}{d\theta}\psi(t), \text{ terminal condition, } C(t_1) = 0, \text{ solved from } t_1 \text{ to}\, t_0
\end{aligned} \tag{19}$$

where C is again used to represent $\frac{dL}{d\theta}$.

The adjoint ODE derived for backpropagation through the TDSE is time-reversible in an analogous manner to the forward Schrödinger dynamics. In fact, under a Hermitian Hamiltonian $H(t)$, the adjoint system is governed by the time-reversed Schrödinger equation. Consider the forward TDSE:

$$\frac{\partial \psi}{\partial t} = -iH\psi, \qquad \psi(0) = \psi_0, \tag{20}$$

with formal solution $\psi(T) = U(0,T)\,\psi_0$, where $U(0,T)$ is the unitary time-evolution operator. Its inverse is the backward evolution $U(0,T)^{-1} = U(0,T)^\dagger = U(T,0)$. If L is a loss depending on the final state $\psi(T)$, the adjoint state $a(t) \equiv \partial L/\partial\psi(t)$ evolves from $t = T$ back to $t = 0$ according to the adjoint ODE:

$$\frac{d}{dt}a(t) = -a(t)\,\frac{\partial f}{\partial \psi}(t) = -a(t)\,(-i\,H(t)) = i\,a(t)\,H(t). \tag{21}$$

This is equivalent to:

$$i\,\frac{\partial \psi}{\partial t}a(t) = -\,H(t)\,a(t), \qquad a(T) = \frac{\partial L}{\partial \psi(T)}, \tag{22}$$

which is the time-reversed TDSE. Its solution is $a(t) = U(t,T)^\dagger\,a(T)$, confirming that backward integration of the adjoint recovers exact gradient information, and establishing analytical time-reversibility.

3.2 Structure-preserving ODE solvers

To simulate the evolution of the wavefunction ψ under the Schrödinger equation, we use the Crank–Nicolson method, a time-stepping scheme that is both second-order accurate and unconditionally stable. It is a finite difference method that averages the explicit (forward Euler) and implicit (backward Euler) time discretizations. For the TDSE, this leads to the update rule:

$$\left(I+\frac{i\Delta t}{2}H\right)\psi^{n+1}=\left(I-\frac{i\Delta t}{2}H\right)\psi^{n}, \tag{23}$$

where ψ^n denotes the wavefunction at time step n. For ease of notation, we will rewrite this as

$$A\psi^{n+1}=B\psi^{n}. \tag{24}$$

Let

$$A=I+\frac{i\Delta t}{2}H,\qquad B=I-\frac{i\Delta t}{2}H,\qquad U(\Delta t)=A^{-1}B, \tag{25}$$

be the one–step Crank–Nicolson update for a time-independent, Hermitian Hamiltonian H.

Theorem 3 (Time-reversibility of Crank–Nicolson)**.** *For every Δt, we have*

$$U(-\Delta t)\,U(\Delta t)=I. \tag{26}$$

Hence propagating a state forward by one step and then backward by the same step returns the original wave-function.

Proof. Because

$$U(-\Delta t)=B^{-1}A, \tag{27}$$

we have

$$U(-\Delta t)\,U(\Delta t)=B^{-1}A\,A^{-1}B=B^{-1}B=I. \tag{28}$$

□

Theorem 4 (Norm conservation of Crank–Nicolson)**.** *The Crank–Nicolson update operator $U(\Delta t)=A^{-1}B$ is unitary:*

$$U(\Delta t)^{\dagger}\,U(\Delta t)=I. \tag{29}$$

Proof. Hermiticity of H gives

$$A^{\dagger}=B,\qquad B^{\dagger}=A. \tag{30}$$

Hence

$$U(\Delta t)^{\dagger}=(A^{-1}B)^{\dagger}=B^{\dagger}\,(A^{\dagger})^{-1}=A\,B^{-1}, \tag{31}$$

and therefore

$$U(\Delta t)^{\dagger}\,U(\Delta t)=A\,B^{-1}A^{-1}B=B\,B^{-1}A^{-1}A=I, \tag{32}$$

because A and B commute since both A and B are just I plus (or minus) a constant times the same matrix H. Thus, the discrete evolution conserves the L^2 norm exactly. □

4 Training workflow and numerical results

This section presents the architecture and training process of the neural network used to learn the potential $V(x)$, along with the numerical results obtained. The model integrates a neural network predicting $V(x)$ with a custom Crank–Nicolson adjoint solver to evolve wavefunctions and compute gradients efficiently. Training leverages learned Fourier features to enrich spatial inputs and focuses gradient updates on the hardest samples for improved stability. Numerical experiments demonstrate the model's ability to recover the double-well potential structure from limited wavefunction pairs, with analysis of its strengths and limitations, including sensitivity to potential parameters and suggestions for future improvements.

4.1 Neural network architecture and training process

The learning process of this model involves two main components, the neural network that predicts $V(x)$ and its training loop, and the custom Crank–Nicolson adjoint solver. First, a neural network learns to approximate a mapping from spatial coordinates to potential energy values based on training data. Input positions are first transformed via learned Fourier features to capture spatial frequencies more effectively, giving the neural net a richer input. The network is trained on pairs of initial and evolved wavefunctions sampled from the dataset. For each training sample, the initial wavefunction is numerically evolved forward in time using the Crank–Nicolson solver driven by the current neural potential estimate. The discrepancy between the predicted and true wavefunction at the final time step is quantified using a fidelity-based loss. To improve training stability, the optimizer updates parameters using gradients aggregated from the hardest samples in each epoch. Then, the core numerical engine evolves the complex wavefunction over discretized time steps with the Crank–Nicolson implicit scheme, ensuring stability and norm preservation. Spatial discretization results in a sparse linear system involving the Hamiltonian operator, constructed from the learned potential and kinetic terms. Forward integration stores the full time evolution of the wavefunction. For backpropagation, the adjoint method solves a related linear system backward in time using the same Crank–Nicolson scheme to compute gradients of the loss with respect to the neural potential parameters. This avoids storing intermediate gradients explicitly and enables memory-efficient training of the network.

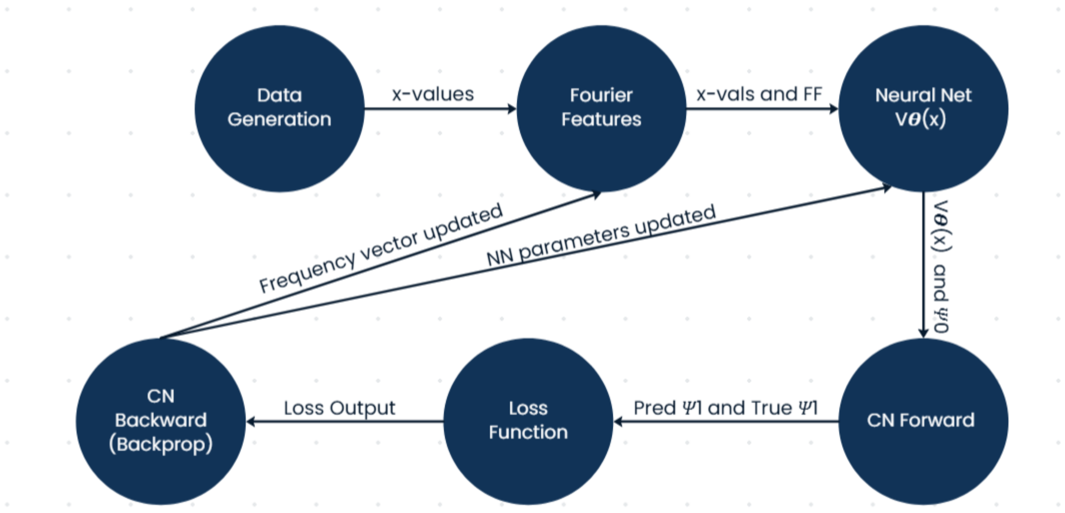

Figure 1: Flowchart of model training process. The full code is available here: GitHub Repository.

4.2 Numerical results

Figure 2 displays the results of the model after training on 250 data pairs of ψ_0 and ψ_1, which were generated using a double-well potential with the form

$$V(x) = a(x^2 - b)^2 \tag{33}$$

and $a = 20, b = 0.5$. Each ψ_0 is a Gaussian with a random mean between $[-0.9, 0.9]$ and random variance between $[0.04, 1]$. Crank–Nicolson is applied over 50 time steps in order to evolve ψ_0 into ψ_1 and the spatial grid contains 500 points. In this analysis, loss adjustment was performed using only the top $k = 20$ predicted ψ_1 instances exhibiting the highest individual losses, instead of the full set. Additionally, the Fourier Features module is set to one frequency, to prevent the predicted $V(x)$ from becoming too oscillatory. On an NVIDIA 4070 GPU, this training process takes about 12 minutes in order to see the results displayed in the figure, which was run for 180 epochs.

The double-well structure in the learned potential is clearly visible, though it appears to be shifted upward by a constant, which does not affect the evolution of ψ_1. However, some discrepancies remain: the wells in the predicted potential are not as pronounced as in the true $V(x)$, and the final wavefunction appears vertically shifted compared to the ground truth ψ_1 from the dataset. The success of this model also seems sensitive to the initial inputs, as it does not perform as well with predicting alternate $V(x)$ functions such as the single well, and its success also relies on the parameters a and b. Since the a parameter of the double-well potential determines how deep the wells are, when its value is increased, the network struggles to achieve similar deep wells. Further experimentation with the loss function may lead to improved performance, as combining standard loss terms such as fidelity loss and mean squared error, could encourage the network to produce predictions more closely aligned with the true ψ_1.

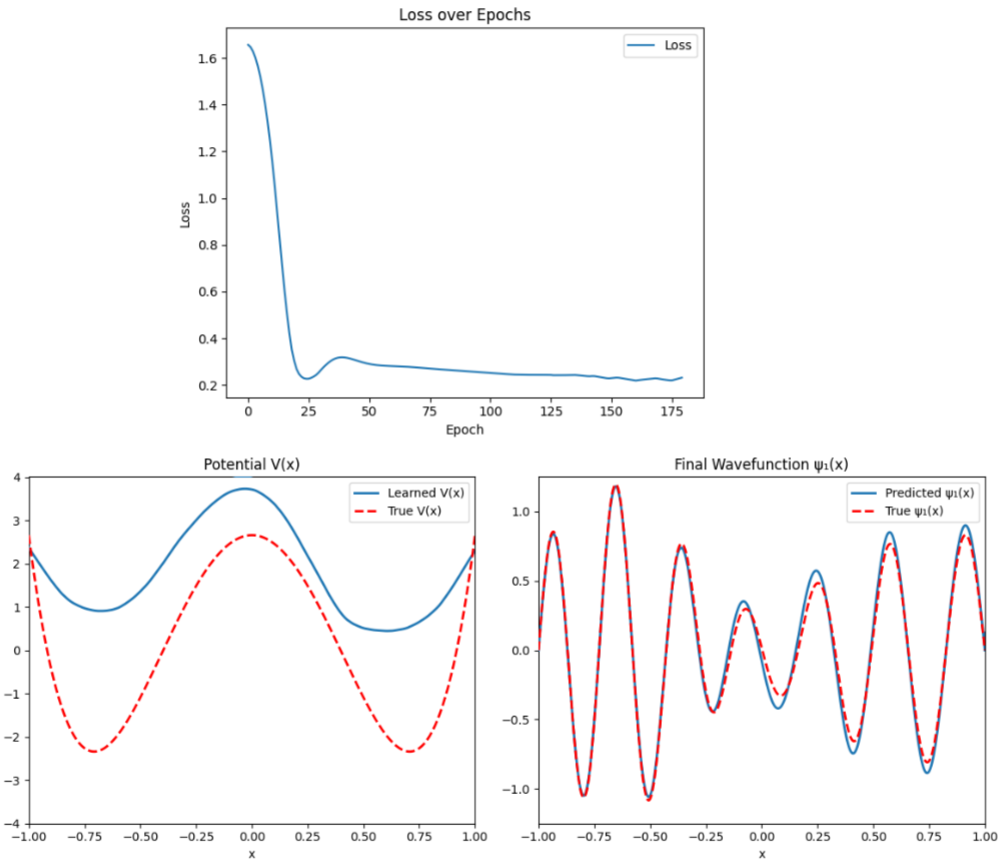

Figure 2: **Results:** The model was trained for 180 epochs using 250 wavefunction pairs generated with a double-well potential defined by $V(x) = 20(x^2 - 0.5)^2$. The result shows that the model successfully captures the double-well structure.
(Top) Loss over Epochs: This plot shows the model's training loss decreasing over 180 epochs, indicating that the model is learning successfully. The x-axis represents the training epoch, and the y-axis is the calculated loss.
(Bottom-Right) Potential V(x): This panel compares the learned potential to the true double-well potential. The x-axis is the spatial position x, and the y-axis is the potential energy $V(x)$.
(Bottom -Left) Final Wavefunction $\psi_1(x)$: This plot shows the real and imaginary components of the final predicted wavefunction versus the true final wavefunction. The x-axis represents the spatial position x, and the y-axis is the amplitude of the wavefunction's components.

5 Summary and outlook

In this project, we addressed the inverse problem of recovering an unknown potential $V(x)$ in the TDSE using limited observations of a quantum system's evolution. Specifically, we used pairs of wavefunctions ψ_0 and ψ_1 that represent the quantum state at two separate times. This type of problem is known to be ill-posed, as many different potentials can result in similar quantum behavior, and small changes in the input can significantly affect the inferred solution [7]. Nevertheless, our goal was to determine whether a machine learning model that respects the physics of the system could accurately infer such potentials from sparse data. To accomplish this, we designed a structure-preserving neural ODE framework in which a neural network parameterizes the potential $V(x)$ and is embedded into a solver for the TDSE. The Crank–Nicolson method was used to evolve the wavefunction over time while maintaining probability conservation and time-reversibility [6]. To efficiently compute gradients without storing the entire trajectory, we applied the adjoint sensitivity method adapted to the TDSE [5]. The neural network was trained by minimizing the discrepancy between the predicted and ground truth wavefunctions at the final time. To enhance spatial expressiveness, we used learned Fourier feature mappings [8], and to improve stability, the loss was computed using only the highest-error samples from each batch.

The model showed promising results when trained on 250 synthetic wavefunction pairs derived from a known double-well potential. It successfully recovered the overall structure of the true potential and remained robust on clean test data. However, it sometimes struggled to reconstruct deep or sharply varying wells, indicating areas for further refinement. While the current setup demonstrates that a neural network can successfully infer a potential function from wavefunction evolution data, several steps could be taken to make the project more realistic and extend its applicability. In experimental settings, the full time evolution of a quantum wavefunction is rarely observable. Instead, one typically has access to only a few time slices, often just the initial and final states [9]. To better reflect this constraint, future versions of the model could be trained using only partial observations of ψ, such as just ψ_0 and ψ_1, or sparse intermediate samples. This would more closely simulate real-world data acquisition and challenge the model to generalize from limited information.

Additionally, the dataset used in this project was generated entirely from a single double-well potential. A more realistic scenario would involve wavefunction pairs generated from a variety of underlying potentials, such as wells, barriers, and disordered landscapes. Training the model on such a dataset would allow it not only to infer the shape of the potential but also to implicitly classify or recognize which potential form gave rise to the observed wavefunction pair. This would make the framework more powerful for tasks like quantum system identification or experimental inference.

Another important step would be the incorporation of physical constraints and robustness to noise. In practical applications, data may be noisy, and models that enforce properties such as norm preservation, boundary behavior, or smoothness of the potential would be more stable and physically grounded. These constraints could be integrated directly into the neural network architecture or added as regularization terms during training.

Finally, the model could be extended to handle time-dependent potentials $V(x,t)$ rather than assuming a static potential throughout the evolution. This would significantly broaden the range of physical systems that can be modeled, including situations where external fields or interactions vary over time.

Overall, the current project provides a strong foundation for potential learning using neural ODEs, and with further development, it could evolve into a versatile framework for interpreting quantum dynamical data in more realistic and experimentally relevant settings.

6 Acknowledgments

We thank our mentor, Professor Weiqi Chu, for her mentoring and support throughout the Research Experience for Undergraduates. We also extend our thanks to the National Science Foundation (NSF) for partially supporting this project under grant DMS-2309814.

References

[1] David Griffiths. *Introduction to Quantum Mechanics.* Cambridge University Press, 1995.

[2] Matthias Kling and Marc Vrakking. Attosecond Electron Dynamics. *Annual review of physical chemistry*, 59:463–92, 02 2008.

[3] M F Ciappina, J A Pérez-Hernández, A S Landsman, W A Okell, S Zherebtsov, B Förg, J Schötz, L Seiffert, T Fennel, T Shaaran, T Zimmermann, A Chacón, R Guichard, A Zaïr, J W G Tisch, J P Marangos, T Witting, A Braun, S A Maier, L Roso, M Krüger, P Hommelhoff, M F Kling, F Krausz, and M Lewenstein. Attosecond Physics at the Nanoscale. *Reports on Progress in Physics*, 80(5):054401, March 2017.

[4] Yann LeCun, Y. Bengio, and Geoffrey Hinton. Deep Learning. *Nature*, 521:436–44, 05 2015.

[5] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural Ordinary Differential Equations. *Advances in Neural Information Processing Systems*, 31, 2018.

[6] Alberto Castro, Miguel AL Marques, and Angel Rubio. Propagators for the Time-Dependent Kohn–Sham Equations. *The Journal of Chemical Physics*, 121(8):3425–3433, 2004.

[7] Yiran Wang and Zhen Li. Inverse Problem of Nonlinear Schrödinger Equation as Learning of Convolutional Neural Network. 2021.

[8] Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[9] Debarshi Kundu, Avimita Chatterjee, Archisman Ghosh, and Swaroop Ghosh. Capturing Quantum Snapshots from a Single Copy via Mid-Circuit Measurement and Dynamic Circuit. *arXiv preprint arXiv:2504.21250*, 2025.