Physics-Informed Neural Networks as Solvers for the Time-Dependent Schrödinger Equation

Karan Shah
Center for Advanced Systems Understanding
Helmholtz-Zentrum Dresden-Rossendorf
Görlitz, Germany
k.shah@hzdr.de

Patrick Stiller
Institute of Radiation Physics
Helmholtz-Zentrum Dresden-Rossendorf
Dresden, Germany
p.stiller@hzdr.de

Nico Hoffmann
Institute of Radiation Physics
Helmholtz-Zentrum Dresden-Rossendorf
Dresden, Germany
n.hoffmann@hzdr.de

Attila Cangi
Center for Advanced Systems Understanding
Helmholtz-Zentrum Dresden-Rossendorf
Görlitz, Germany
a.cangi@hzdr.de

Abstract

We demonstrate the utility of physics-informed neural networks (PINNs) as solvers for the non-relativistic, time-dependent Schrödinger equation. We study the performance and generalisability of PINN solvers on the time evolution of a quantum harmonic oscillator across varying system parameters, domains, and energy states.

1 Introduction

In recent years, there was a surge in the use of machine learning (ML) techniques in the physical sciences [1], referred to as scientific machine learning. This has led to the rise of physics-informed machine learning [2], where physics-based constraints are used to guide ML models, achieved by incorporating structured prior information derived from physical laws into the learning algorithm.

Given the importance of numerical simulations across scientific disciplines, many ML surrogates for solving differential equations on a large scale [3] have been developed. These approaches include the deep Galerkin method [4], PINNs [5], neural operators [6], and DeepONets [7].

PINNs are a class of ML algorithms for solving forward and inverse problems that are represented by partial differential equations. As opposed to numerical solvers where a new solution must be computed whenever there is a change in the domain or system parameters, PINNs offer a mesh-free alternative where the solver can be used for systems on arbitrary grid resolutions and system parameters at a fixed inference cost [8]. While training the PINN models can be a substantial computational cost initially, it can be amortized over time due to rapid inference over a wider set of system parameters. However, PINNs also have demonstrated shortcomings such as spectral bias against high-frequency solutions [9], overfitting to trivial solutions [10], and poor performance for systems with shocks [11] and for larger time domains [12]. The theory and applications of PINNs are active areas of research, with developments such as the RNN-DCT-PINN [12], the gated-PINN [13], and the variational-PINN [14] which increase their performance and scalability. Popular ML frameworks for PINN models include Neural Solvers [13], Modulus [15], and DeepXDE [16].

The non-relativistic Schrödinger equation is the fundamental equation for describing quantum systems. The wavefunction ϕ, from which all observables of a system can be calculated, is obtained by solving it. We distinguish two classes of problems: the time-independent Schrödinger equation (TISE) yields the wave function ϕ and the associated energies for a static system in terms of an eigenvalue problem,

while the time-dependent Schrödinger equation (TDSE) describes the dynamics of a quantum system, i.e., the time evolution of ϕ. Prior works have dealt with using ML to solve the SE. These include models such as fully-connected networks (FCN) [17, 18], reservoir computing [19], PINNs [20] for the eigenvalue problem defined by the TISE, and residual networks [21] and LSTMs [22] for the TDSE. The TDSE has also been used as benchmark for PINNs [5, 2, 13]. However, in this work, we go beyond prior investigations by examining the utility of PINN solvers for the TDSE across varying system parameters, domains, and energy states.

2 Methods

2.1 PINNs

PINNs are constructed by encoding the constraints posed by a given differential equation and its boundary conditions into the loss function of a deep learning model, usually, an FCN. These constraints guide the network to finding a solution to the differential equation.

A partial differential equation (PDE) is defined by f with solution $u(\mathbf{x}, t)$ governed by

$$f(u) := u_t + \mathcal{N}[u; \lambda], \ \boldsymbol{x} \in \Omega, \ t \in [0, T], \qquad f(u) = 0, \tag{1}$$

where $\mathcal{N}[u; \lambda]$ is a differential operator parameterised by λ, $\Omega \in \mathbb{R}^{\mathbb{D}}$, and $\boldsymbol{x} = (x_1, x_2, ..., x_d)$ with boundary conditions $\mathcal{B}(u, \boldsymbol{x}, t) = 0$ on $\partial\Omega$ and initial conditions $\mathcal{T}(u, \boldsymbol{x}, t) = 0$ at $t = 0$.

A neural network $u_{net} : \mathbb{R}^{D+1} \mapsto \mathbb{R}^1$ is constructed as a surrogate model $f_{net} = f(u_{net})$ for the true solution u. Constraints are encoded in the loss term L for neural network optimization

$$L = \lambda_f L_f + \lambda_{BC} L_{BC} + \lambda_{IC} L_{IC}, \tag{2}$$

with $\lambda_f, \lambda_{BC}, \lambda_{IC}$ being the regularization parameters. The PDE loss $L_f = (1/N_f) \sum_{i=1}^{N_f} \left| f_{net} \left(\boldsymbol{x}_f^i, t_f^i \right) \right|^2$ denotes the error in the solution within the interior points of the system and is calculated for N_f collocation points. The boundary loss $L_{BC} = (1/N_{BC}) \sum_{i=1}^{N_{BC}} \left| u_{net} \left(\boldsymbol{x}_{BC}^i, t_{BC}^i \right) - u \left(\boldsymbol{x}_{BC}^i, t_{BC}^i \right) \right|^2$ is the constraint imposed by the boundary conditions, whereas $L_{IC} = (1/N_{IC}) \sum_{i=1}^{N_{IC}} \left| u_{net} \left(\boldsymbol{x}_{IC}^i, t_{IC}^i \right) - u \left(\boldsymbol{x}_{IC}^i, t_{IC}^i \right) \right|^2$ imposes the initial conditions. Both of them are calculated on a set of N_{BC} boundary points and N_{IC} initial points, respectively, where u_{net} refers to the approximate solution predicted by the PINN and u denotes the true value according to the boundary and initial conditions at that point. The distribution of collocation points is visualized in Fig. 5b.

Once trained, the neural network is used to solve the PDE, potentially for a range of parameters λ [8].

2.2 Time Dependent Schrödinger Equation

A PINN is constructed for solving the TDSE

$$i\frac{\partial \phi(\mathbf{r}, t)}{\partial t} - \hat{H}\phi(\mathbf{r}, t) = 0, \tag{3}$$

where $\hat{H}$ denotes the Hamiltonian of the problem and $\phi(\mathbf{r}, t)$ the solution which is commonly referred to as a wave function that depends on a spatial coordinate $\mathbf{r}$ in three-dimensional space and time t. Note that we adopt Hartree atomic units, i.e., $\hbar = k_{\text{B}} = m_e = 1$, so energies are measured in Hartree and lengths in Bohr radii. The Hamiltonian represents the specific problem, such as the kinetic and potential energies of particle species and their interactions among each other and with external fields.

In this conceptual work, we resort to a simple but archetypical model problem, namely non-interacting and spinless particles in a quantum harmonic oscillator in one spatial dimension with Hamiltonian

$$\hat{H}_x = -\frac{1}{2}\frac{\partial^2}{\partial x^2} + \frac{\omega^2}{2}x^2, \tag{4}$$

where ω denotes the frequency of the harmonic oscillator. The most fundamental kind of quantum dynamics is achieved by a superposition of eigenstates which are solutions of the corresponding TISE. The analytical eigenstates of the harmonic oscillator are $\phi_n(x) = \phi_0(x)(2^n n!)^{-1/2} H_n(\sqrt{\omega}x)$

and the corresponding eigenvalues are $\epsilon_n = \omega(n + 1/2)$ with $n \in \mathbb{N}$, where H_n denotes the Hermite polynomials and $\phi_0(x) = (\omega/\pi)^{1/4} \exp\left(-\omega x^2/2\right)$. Superimposing two eigenstates $\phi_{m,n}(x,t) = \left[e^{-i\epsilon_m t}\phi_m(x) + e^{-i\epsilon_n t}\phi_n(x)\right]/\sqrt{2}$ yields a solution of the TDSE, where the entire dynamics takes place due to the phases $\exp\left(-i\epsilon_n t\right)$. Since the neural network is constrained to $\mathbb{R}$, the complex-valued solutions can be represented as $\phi(x,t) = u + iv$, where $u = \mathrm{Re}(\phi)$ is the real part and $v = \mathrm{Im}(\phi)$ the imaginary part. The TDSE can then be written as in terms of u and v as

$$\left(-\frac{\partial v}{\partial t} + \frac{1}{2}\frac{\partial^2 u}{\partial x^2} - \frac{\omega^2}{2}x^2\right) + i\left(\frac{\partial u}{\partial t} + \frac{1}{2}\frac{\partial^2 v}{\partial x^2} - \frac{\omega^2}{2}x^2\right) = 0 \,. \tag{5}$$

Consequently, a PINN was constructed with inputs (x, t, ω) and outputs (u_{net}, v_{net}). The error of the predictions was quantified on the probability density $|\phi(x,t)|^2 = u(x,t)^2 + v(x,t)^2$.

3 Results

The domains for the TDSE of the quantum harmonic oscillator are $x \in [-\pi, \pi]$ and $t \in [0, 2\pi]$ with Dirichlet boundary conditions $u(x_0, t) = 0$, $v(x_0, t) = 0$ for $x_0 \in \{-\pi, \pi\}$ and initial conditions $u(x, 0) = \mathrm{Re}\left[\phi_{m,n}(x, 0)\right] = \phi_{m,n}(x, 0)$, $v(x, 0) = \mathrm{Im}\left[\phi_{m,n}(x, 0)\right] = 0$.

Table 1: Results. (Here FP denotes FCN-PINN.)

System	Solver	MSE $|\phi|^2$	Training time (min)	Inference time (s)
Baseline	FP	2.94e-5	16.12	4.26
Generalisability	FP (Interpolation)	(2.72±2.18)e-5	130.72	(9.79±0.10)
	FP (Extrapolation)	(2.65±1.56)e-3		
Larger time domain	FP	4.38e-2	57.56	12.25
	FP (Causal)	3.62e-3	63.33	12.35
Higher energy states	FP	1.25e-2	29.73	3.98
	FP (Causal)	1.49e-3	85.85	4.28

3.1 Baseline

The initial surrogate model was created for a one-dimensional quantum harmonic oscillator with a fixed parameter $\omega = 1$. The initial state was chosen as a superposition of the ground and the first excited states $\phi_{0,1}(x,t) = \left[e^{-i\epsilon_0 t}\phi_0(x) + e^{-i\epsilon_1 t}\phi_1(x)\right]/\sqrt{2}$.
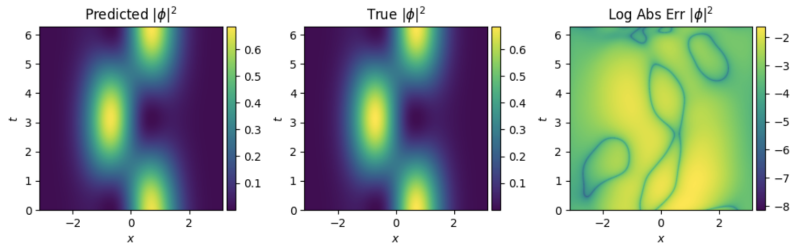

Figure 1: True and predicted values for density $|\phi|^2$ for $\phi_{0,1}(x,t)$ with $\omega = 1.0$.

Fig. 1 shows the results for the baseline model. It illustrates how the probability density of the superpositions state propagates in time, where the left plot shows the prediction of the PINN and the center plot the ground truth. As shown, on the right plot, the PINN solver yields the structure of the solution very accurately. The spatially and temporally resolved MSEs are in the order of 10^{-5}.
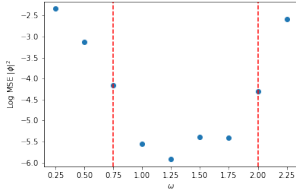
3

Figure 2: Log MSE $|\phi|^2$ for PINN solutions over varying values of ω. Training range within red dashed lines.
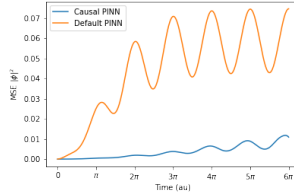

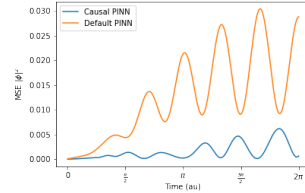Figure 3: MSE $|\phi|^2$ over time for large time domain.

Figure 4: MSE $|\phi|^2$ over time for higher energy states.

3.2 Generalisability

To demonstrate the performance over changing parameters, the PINN was trained for the same initial state as the baseline in a range of values $\omega \in [0.75, 2.0]$. After training, the inference was carried out for $\omega \in [0.5, 2.5]$. As illustrated in Fig. 2, good performance was observed on previously unseen ω values within the range of the training range (interpolation) but accuracy diminished somewhat as the values of ω deviated from the training range during inference (extrapolation). This can be attributed to the change in the width of the wave function relative to our fixed space domain. The width is inversely proportional to ω. For lower values of ω, the width is too broad and violates the fixed domain boundary conditions. For higher values of ω, the width is too narrow and the network underfits to a zero-valued solution. This problem could be alleviated by using dynamic domains while training PINNs, such that the domain adjusts to the width of the wave function.

3.3 Larger time domains

Next, a longer time domain $t \in [0, 6\pi]$ for the baseline $\phi_{0,1}(x, t)$ system was studied. The FCN-PINN resulted in a high mean error in the order of 10^{-2} with a loss of structure at later time steps as shown in Fig. 3 (orange curve). To mitigate this, the loss function was modified to take causality into account [23]. This was done by splitting the time domain into equally spaced components, with the loss for each component being a weighted cumulative sum of L_f of the domains up to that component. This enforces causality in the time domain. With this modification, the error is reduced to 10^{-3} with the correct structure preserved throughout the time domain (blue curve).

3.4 Higher energy states

As a demonstration for higher energy states, the FCN-PINN was trained on the initial state $\phi_{0,3}(x, t) = \left[e^{-i\epsilon_0 t} \phi_0(x) + e^{-i\epsilon_3 t} \phi_3(x) \right] / \sqrt{2}$. The predicted solution with the default FCN architecture falsely converged to the static ground state ϕ_0 as time evolved. As can be seen from Fig. 4, the error increases significantly for later time steps (orange curve). Subsequently, a PINN with a loss function that preserves causality, as explained in Section 3.3, was used. This solves the problem of false convergence. It captures the structure of the solution over a long period of time evolution with a low MSE on the order of 10^{-3} (blue curve).

4 Future Work

There has been a rise in the applications of machine learning in computational chemistry and materials science [24]. Density functional theory (DFT) is the most widely used method for computing chemical and material properties. Hence, DFT calculations utilize a considerable proportion of computational resources on academic HPC systems. The underlying equations for time-dependent DFT are the Schrödinger-like time-dependent Kohn-Sham (TDKS) equations:

$$\left[-\frac{1}{2}\nabla^2 + v_s[n](\boldsymbol{r}, t) \right] \phi_j(\boldsymbol{r}, t) = i\frac{\partial \phi_j(\boldsymbol{r}, t)}{\partial t} \tag{6}$$

where the electronic density $n(\boldsymbol{r}, t) = \sum_j |\phi_j(\boldsymbol{r}, t)|^2$ is the quantity of interest. The Kohn-Sham potential $v_s[n](\boldsymbol{r}, t)$ is a functional of the density n which is a function of position $\boldsymbol{r}$ and time t. These coupled equations are solved through iterative numerical algorithms which represent a significant computational cost in the TDDFT workflow[25, 26]. Given the effectiveness of PINNs for the TDSE, the next step is to use PINNs to accelerate TDKS calculations.

5 Conclusion

PINNs act as surrogate models for solving the TDSE. The primary advantage of PINNs over traditional numerical solvers is that they are mesh-free and can be used to perform simulations at arbitrary resolutions once sufficiently trained. Depending upon the training regiment, PINNs can be generalized to a range of the PDE parameter space. Their mesh-free nature and generalisability can be utilized for accelerating simulations of electron dynamics. While FCN-PINNs are not sufficient to reproduce complex dynamics, this can be alleviated with techniques such as causal training. Extending this work to TDDFT, a PINN framework would enable on-the-fly modeling of the electronic response properties of laser-excited or shock-compressed samples in various scattering experiments that are conducted at photon sources around the globe. This would enable fast simulations that generalize well over the input parameters of the experimental setup.

6 Broader Impact

While this work is at a preliminary stage, we believe that accelerating TDDFT calculations is a net benefit. Accelerating TDDFT calculations has a broad impact on the fields such as computational chemistry and materials science with applications in myriad domains such as materials design, drug discovery, and green energy generation. Results obtained in this line of research can also be used to accelerate numerical simulations in other fields of science and engineering.

References

[1] Giuseppe Carleo, Ignacio Cirac, Kyle Cranmer, Laurent Daudet, Maria Schuld, Naftali Tishby, Leslie Vogt-Maranto, and Lenka Zdeborová. Machine learning and the physical sciences. *Reviews of Modern Physics*, 91(4):045002, December 2019.

[2] George Em Karniadakis, Ioannis G. Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440, June 2021.

[3] Jan Blechschmidt and Oliver G. Ernst. Three ways to solve partial differential equations with neural networks — A review. *GAMM-Mitteilungen*, 44(2):e202100006, 2021.

[4] Justin Sirignano and Konstantinos Spiliopoulos. DGM: A deep learning algorithm for solving partial differential equations. *Journal of Computational Physics*, 375:1339–1364, December 2018.

[5] 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*, 378:686–707, February 2019.

[6] Zongyi Li, Nikola Borislavov Kovachki, Kamyar Azizzadenesheli, Burigede liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations*, 2021.

[7] Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3(3):218–229, March 2021.

[8] Stefano Markidis. The old and the new: Can physics-informed deep-learning replace traditional linear solvers? *Frontiers in Big Data*, 4, 2021.

[9] Sifan Wang, Xinling Yu, and Paris Perdikaris. When and why PINNs fail to train: A neural tangent kernel perspective. *arXiv:2007.14527 [cs, math, stat]*, July 2020.

[10] Raphael Leiteritz and Dirk Pflüger. How to Avoid Trivial Solutions in Physics-Informed Neural Networks. *arXiv:2112.05620 [cs, stat]*, December 2021.

[11] Olga Fuks and Hamdi A. Tchelepi. Limitations of physics informed machine learning for nonlinear two-phase transport in porous media. *Journal of Machine Learning for Modeling and Computing*, 1(1):19–37, 2020.

[12] Benjamin Wu, Oliver Hennigh, Jan Kautz, Sanjay Choudhry, and Wonmin Byeon. Physics informed RNN-DCT networks for time-dependent partial differential equations. In Derek Groen, Clélia de Mulatier, Maciej Paszynski, Valeria V. Krzhizhanovskaya, Jack J. Dongarra, and Peter M. A. Sloot, editors, *Computational Science – ICCS 2022*, pages 372–379, Cham, 2022. Springer International Publishing.

[13] Patrick Stiller, Friedrich Bethke, Maximilian Böhme, Richard Pausch, Sunna Torge, Alexander Debus, Jan Vorberger, Michael Bussmann, and Nico Hoffmann. Large-Scale Neural Solvers for Partial Differential Equations. In Jeffrey Nichols, Becky Verastegui, Arthur 'Barney' Maccabe, Oscar Hernandez, Suzanne Parete-Koon, and Theresa Ahearn, editors, *Driving Scientific and Engineering Discoveries Through the Convergence of HPC, Big Data and AI*, Communications in Computer and Information Science, pages 20–34, Cham, 2020. Springer International Publishing.

[14] Ehsan Kharazmi, Z. Zhang, and George Em Karniadakis. Variational physics-informed neural networks for solving partial differential equations. *ArXiv*, abs/1912.00873, 2019.

[15] Oliver Hennigh, Susheela Narasimhan, Mohammad Amin Nabian, Akshay Subramaniam, Kaustubh Tangsali, Max Rietmann, Jose del Aguila Ferrandis, Wonmin Byeon, Zhiwei Fang, and Sanjay Choudhry. NVIDIA SimNet{TM}: An AI-accelerated multi-physics simulation framework. *arXiv:2012.07938 [physics]*, December 2020.

[16] Lu Lu, Xuhui Meng, Zhiping Mao, and George Em Karniadakis. DeepXDE: A deep learning library for solving differential equations. *SIAM Review*, 63(1):208–228, 2021.

[17] I.E. Lagaris, A. Likas, and D.I. Fotiadis. Artificial neural network methods in quantum mechanics. *Computer Physics Communications*, 104(1-3):1–14, August 1997.

[18] A. Radu and C. A. Duque. Neural network approaches for solving Schrödinger equation in arbitrary quantum wells. *Scientific Reports*, 12(1):2535, February 2022.

[19] L. Domingo, J. Borondo, and F. Borondo. Adapting reservoir computing to solve the Schrödinger equation. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 32(6):063111, June 2022.

[20] Henry Jin, Marios Mattheakis, and Pavlos Protopapas. Unsupervised neural networks for quantum eigenvalue problems. In *2020 NeurIPS Workshop on Machine Learning and the Physical Sciences*. NeurIPS, NeurIPS, 2020.

[21] Changjian Xie, Jingrun Chen, and Xiantao Li. A Machine-Learning Method for Time-Dependent Wave Equations over Unbounded Domains, July 2021.

[22] Bin Yang, Mengxi Wu, and Winfried Teizer. Conditional Seq2Seq model for the time-dependent two-level system, June 2022.

[23] Sifan Wang, Shyam Sankaran, and Paris Perdikaris. Respecting causality is all you need for training physics-informed neural networks. *ArXiv*, abs/2203.07404, 2022.

[24] L. Fiedler, K. Shah, M. Bussmann, and A. Cangi. Deep dive into machine learning density functional theory for materials science and chemistry. *Phys. Rev. Materials*, 6:040301, Apr 2022.

[25] Alberto Castro, Miguel A. L. Marques, and Angel Rubio. Propagators for the time-dependent kohn–sham equations. *The Journal of Chemical Physics*, 121(8):3425–3433, 2004.

[26] Adrián Gómez Pueyo, Miguel A. L. Marques, Angel Rubio, and Alberto Castro. Propagators for the Time-Dependent Kohn–Sham Equations: Multistep, Runge–Kutta, Exponential Runge–Kutta, and Commutator Free Magnus Methods. *Journal of Chemical Theory and Computation*, 14(6):3040–3052, June 2018.

[27] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

A   Computation Details

A.1   Network architecture and hyperparameters

A.1.1   Sampling

We used a batch size of 3140 points for the interior, 200 points for the boundary conditions and 314 points for initial conditions.

A.1.2   Architecture

(a) Common architecture of PINNs.

(b) Distribution of collocation points for calculating the loss function in Eq (2).
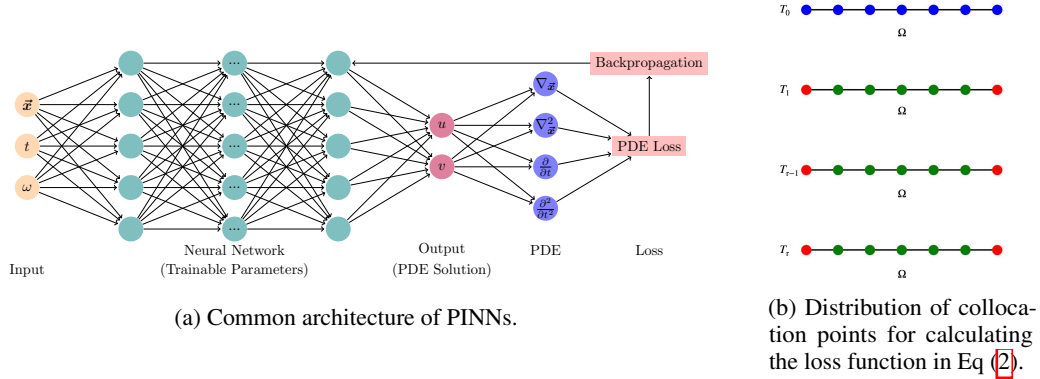
Figure 5: PINN Architecture

Because of the derivatives needed in the PINN loss, the activation function has to be $k+1$ differentiable for a PDE of order k. This restricts the choice of activation functions to functions like tanh and SiLU.

FCN-PINN (Sections 3.1, 3.2, 3.3, 3.4): We used a fully-connected network with 6 layers consisting of 512 neurons each. The activation function used was tanh. We calculate the loss function as described in Eq. 2. We use the ADAM optimizer with $\beta_1 = 0.09, \beta_2 = 0.999$. The learning rate is initialized at $\alpha_0 = 0.001$ with exponential decay rate $\gamma = 0.9$ at decay steps $t_\gamma = 2000$ training steps with schedule $\alpha_t = \alpha_0 \gamma^{\frac{t}{t_\gamma}}$.

For the generalisability study (Section 3.2), we used a fully-connected network with 12 layers consisting of 512 neurons each.

Causal-PINN (Sections 3.3, 3.4): We used a fully-connected network with 6 layers consisting of 512 neurons each. The activation function used was SiLU. We calculate the loss function with the causal training scheme described in [23]. We use the ADAM optimizer with $\beta_1 = 0.09, \beta_2 = 0.999$. The learning rate is initialized at $\alpha_0 = 0.001$ with exponential decay rate $\gamma = 0.95$ at decay steps $t_\gamma = 100$ training steps with schedule $\alpha_t = \alpha_0 \gamma^{\frac{t}{t_\gamma}}$.

In both cases, training is continued till convergence is reached for MSE $|\phi|^2$ or maximum number of training iterations for each experiment.

A.1.3 Evaluation

The PINN solution $u_{net}(x, t), v_{net}(x, t)$ is used to calculate the density $|\phi_{net}(x, t)|^2 = u_{net}^2(x, t) + v_{net}^2(x, t)$ on a grid with resolution $\Delta x, \Delta t = 0.01$au in the domain specified for that system, i.e. 628×628 uniformly-spaced points in Sections 3.1, 3.2, 3.4 and 1884×628 points in Section 3.3.

$$MSE|\phi_{net}(x, t)|^2 = \frac{1}{N} \sum_{i=1}^{N} \left| |\phi(x_i, t_i)|^2 - |\phi_{net}(x_i, t_i)|^2 \right|^2 , \tag{7}$$

calculated over all N points of the evaluation grid, where $\phi(x_i, t_i)$ is the analytical solution.

A.2 Software and Hardware

A.2.1 Software

We utilise the Modulus PINN framework [15] and PyTorch ML framework [27] in this work.

A.2.2 Hardware

All experiments were carried out on nodes consisting of one 12 Core Intel Xeon 3.0 GHz CPU and one Nvidia Tesla V100 (32 GB) GPU.

HPC Usage: We estimate total usage to be 18 GPU-hours for the experiments in this paper.

B Analytical Solutions

The analytical solution for the baseline is

$$\phi_{0,1}(x, t) = \frac{1}{\sqrt{2}} \sqrt[4]{\frac{\omega}{\pi}} \exp\left(-\frac{\omega x^2}{2}\right) \left(exp\left(-i\frac{\omega}{2}t\right) + exp\left(-i\frac{3\omega}{2}t\right)\sqrt{2\omega}x\right). \tag{8}$$

The analytical solution in higher energy state is

$$\phi_{0,3}(x, t) = \frac{1}{\sqrt{2}} \sqrt[4]{\frac{\omega}{\pi}} \exp\left(-\frac{\omega x^2}{2}\right) \left(exp\left(-i\frac{\omega}{2}t\right) + exp\left(-i\frac{7\omega}{2}t\right)\frac{1}{\sqrt{3}}\left(2\sqrt{\omega^3}x^3 - 3\sqrt{\omega}x\right)\right). \tag{9}$$