



PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA
MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH



ABOU-BAKR BELKAID UNIVERSITY - TLEMCCEN

THESIS

Presented to:

FACULTY OF SCIENCE – DEPARTMENT OF PHYSICS

MASTER IN PHYSICS

MASTER IN COMPUTATIONAL PHYSICS

By :

Ms GOURARI Samah

On the topic

Physics-Informed Neural Networks (PINNs) for Forward and Inverse Quantum Problems

Publicly defended on July 1st 2026 in Tlemcen before the jury composed of:

Mr LIAZID Abdelkrim
Mr BOUFATAH M. Reda
Ms. SENOUDI Assia Rachida

Professor
Associate Professor
Associate Professor

University of Tlemcen
University of Tlemcen
University of Tlemcen

President
Supervisor
Examiner

GOURARI/2026

Academic Year: 2025 ~ 2026

Dedication

To my parents, for everything words will never be enough to express.

To my brothers and sisters, my cousin, my aunt and uncle, for their love and for believing in me all along the way.

To my friend, for his endless patience, his quiet strength, and his belief in me on the days I could not believe in myself. For keeping me grounded, and for being my greatest cheerleader throughout this journey.

And to my hamster Peachy, my smallest and most faithful companion. He was there through every difficult grade, every late night, every moment of doubt. His silent presence and quiet comfort were the engine that carried me from my bachelor years all the way to here. He left us before the end of this journey, but his memory never did. This one is for him.



In loving memory.

Acknowledgements

I would like to express my deepest gratitude to my supervisor, Dr. Mohammed Reda Boufatah, whose support extended far beyond the academic scope of this thesis. His tireless investment in this work, his scientific rigor, and his constant moral support made these results possible. He was not only a guide through the research but a steadying presence throughout this chapter of my life. I could not have done this without him.

I would like to thank the members of the jury, Pr. Senoudi and Pr. Liazid, for their time and careful attention to this work, and for more than that. It was Pr. Senoudi who first sparked my interest in numerical methods, and Pr. Liazid whose insights opened the door to physics-informed neural networks. In many ways, this thesis is the direct result of what they put in motion.

My sincere thanks go to the professors of the Department of Physics at Université Abou Bekr Belkaïd, Tlemcen, and in particular to the teaching staff of the Computational Physics master's programme. I am also grateful to the Dr. Bensaid Rahma for encouraging me and pushing me towards success during my Bachelor.

Finally, I want to thank my colleagues in Computational Physics. This year was not only one of intense learning, it was one of the most memorable years of my life. We did not just study together; we grew together, and I carry that with me.

Contents

Dedication	1
Acknowledgements	i
List of Figures	viii
List of Tables	x
0.1 Motivation	1
0.2 Problem Statement	4
0.2.1 Forward Task	4
0.2.2 Inverse Task	5
0.3 Objectives of the Thesis	5
0.4 Thesis Structure	5
1 The Schrödinger Equation: Mathematical Structure and Problem Formulation	7
1.1 From Determinism to Wave Functions	8
1.2 Probabilistic Nature and Wave Function Interpretation	8
1.3 Operators and Physical Observables	10
1.4 Time-Dependent Schrödinger Equation	13
1.5 Time-Independent Schrödinger Equation	14
1.6 Boundary Conditions for the Schrödinger PDE	15
1.7 Self-Adjointness Properties	17

1.8	Examples of Quantum Systems	18
1.9	Variants of the Schrödinger Equation	19
1.10	Forward and Inverse Problems in Quantum Mechanics	20
1.11	Forward (Direct) Problems	21
1.12	Inverse Problems	23
1.13	Summary	25
2	Classical Numerical Solvers for Forward and Inverse Schrödinger Equation	27
2.1	Classical Forward Solver: Finite Difference Method	28
2.1.1	Core Idea	28
2.1.2	Mathematical Foundations	30
2.1.3	Key Concepts for Stability Analysis	37
2.1.4	Algorithmic Framework: Time-Stepping Schemes	40
2.1.5	Limitations of the Finite Difference Method	49
2.2	Classical Inverse Solver: Tikhonov Regularization	51
2.2.1	Core Idea: Problem Formulation and the Forward Operator	51
2.2.2	Mathematical Foundations	54
2.2.3	Tikhonov Regularization	59
2.2.4	Solution Algorithm	62
2.2.5	Limitations of the Classical Inverse Approach	69
2.2.6	Summary and Transition	71
3	Physics-Informed Neural Networks for Forward and Inverse Quantum Problems	72
3.1	From Classical Solvers to Data-Driven Models	72
3.1.1	The Shift in Perspective	72
3.1.2	Machine Learning: Core Concepts	73
3.1.3	Evaluating Models	74

3.2	Neural Networks	75
3.2.1	Constructing a Neural Network in Practice	75
3.2.2	From Mesh to Function: The Role of a Single Neuron	76
3.2.3	From One Neuron to a Deep Network	81
3.2.4	Loss Function, Backpropagation, and Gradient Descent	82
3.3	Physics-Informed Neural Networks	84
3.3.1	From Data to Physics: Loss Function Changing	84
3.3.2	The PDE Residual	85
3.3.3	Initial and Boundary Condition Losses	87
3.3.4	Joint Training on All Three Losses	88
3.4	Inverse Physics-Informed Neural Networks	88
3.4.1	Unknown Parameters	88
3.4.2	The Data Loss Term	89
3.4.3	Two Gradients	90
3.4.4	Output of Training	90
3.4.5	Non-Uniqueness in the Inverse Problem	91
3.5	Summary	93
4	Implementation of Forward Solvers and Results	95
4.1	Physical System, Theoretical Background and Analytical Solution	95
4.1.1	Physical System	95
4.1.2	Theoretical Background	98
4.1.3	Analytical Solution Implementation	99
4.2	Eigenvalue Method	100
4.2.1	Implementation	100
4.2.2	Accuracy and Norm Conservation	100
4.2.3	Computational Cost	102
4.3	Crank-Nicolson Method	103
4.3.1	Implementation	103

4.3.2	Norm Conservation and Unitarity	104
4.3.3	Accuracy	104
4.3.4	Computational Cost	106
4.4	Test 1: Eigenvalue Method vs Crank-Nicolson	106
4.5	Test 2: Baseline PINN	108
4.5.1	PINN Architecture and Training	108
4.5.2	Results	110
4.5.3	Why the Baseline PINN Fails	111
4.6	Test 3: Improved PINN	112
4.6.1	Architectural Improvements	112
4.6.2	Results	115
4.6.3	What Changed and What Didn't	117
4.7	Supplementary Experiment: High-Capacity PINN at $T=2.0$	117
4.7.1	Training Convergence	119
4.7.2	Results at $T=2.0$	120
4.7.3	Why the PINN Fails at Longer Time Horizons	122
4.8	Supplementary Experiment: All Methods at $T=2.0$	123
4.9	Summary: Forward Problem	125
5	Implementation of Inverse Solvers and Results	127
5.1	Problem Setup and Data Generation	127
5.1.1	Problem Setup	127
5.1.2	Data Generation	129
5.2	Tikhonov Regularization	129
5.2.1	Implementation	129
5.3	Inverse PINN	131
5.3.1	Architecture	131
5.3.2	Loss Function and Physical Priors	132
5.3.3	Training Configuration	133

5.4	Initial Comparison: Tikhonov v1 vs Inverse PINN v1	134
5.4.1	Reference Data	134
5.4.2	Tikhonov v1	134
5.4.3	Inverse PINN v1	135
5.4.4	Direct Comparison	136
5.5	Benchmark: Tikhonov v1–v2 vs Inverse PINN v1–v2	137
5.5.1	Variant Configurations	137
5.5.2	Results	139
5.5.3	Analysis	140
5.6	Tikhonov Robustness to Noise	141
5.7	Benchmark v3: Six-Method Comparison	142
5.7.1	New Adjustments	142
5.7.2	Architecture Comparison	144
5.7.3	Results	146
5.7.4	Analysis	147
5.8	Benchmark v4: Final Variants	149
5.8.1	New Adjustments	149
5.8.2	Architecture Comparison	149
5.8.3	Results	150
5.8.4	Analysis	152
5.9	Summary: Inverse Problem	153
6	Conclusion and Perspectives	155
6.1	Conclusion	155
6.2	Perspectives	158

List of Figures

1	PINN publications by field	3
2	Top PINN research fields	3
2.1	Standard Gaussian density	53
3.1	Single-neuron model	77
3.2	Activation functions	80
3.3	Deep-network architecture	82
3.4	Neural-network training loop	84
3.5	Inverse PINN pipeline	91
3.6	Inverse-problem non-uniqueness	92
4.1	Eigenmethod density snapshots	101
4.2	Eigenmethod density evolution	102
4.3	Crank–Nicolson density snapshots	105
4.4	Crank–Nicolson density evolution	105
4.5	Eigenmethod vs Crank–Nicolson	106
4.6	Baseline PINN comparison	110
4.7	Improved PINN comparison	115
4.8	GPU PINN training history	119
4.9	GPU PINN validation loss	120
4.10	GPU PINN density evolution	120
4.11	GPU PINN wave-function comparison	121

4.12 GPU PINN accuracy metrics	121
4.13 All forward methods comparison	124
5.1 Inverse-problem reference data	134
5.2 Tikhonov v1 reconstruction	134
5.3 Inverse PINN v1 results	135
5.4 Tikhonov v1 vs PINN v1	136
5.5 Four-variant inverse benchmark	139
5.6 Four-variant reconstruction comparison	139
5.7 Tikhonov noise robustness	142
5.8 Six-variant inverse benchmark	146
5.9 Six-variant reconstruction comparison	146
5.10 PINN v3 training history	147
5.11 v4 inverse benchmark	150
5.12 v4 reconstruction comparison	151
5.13 Tikhonov v4 CV map	151
5.14 PINN v4 diagnostics	152

List of Tables

2.1	Time-stepping scheme comparison	49
2.2	Tikhonov inverse algorithm	68
3.1	Standard NN vs PINN	85
4.1	Forward-problem parameters	96
4.2	Discrete eigenvalue accuracy	101
4.3	Test 1 metrics	108
4.4	Baseline PINN configuration	109
4.5	Test 2 metrics	111
4.6	Improved PINN configuration	114
4.7	Test 3 metrics	116
4.8	PINN architecture comparison	118
4.9	GPU PINN metrics	122
4.10	PINN norm-drift comparison	123
4.11	All forward methods metrics	125
5.1	Forward vs inverse parameters	128
5.2	Tikhonov variant summary	131
5.3	Inverse PINN v1 configuration	133
5.4	Initial inverse-method comparison	136
5.5	v1 to v2 configuration changes	138
5.6	Four-variant benchmark results	140

5.7	Tikhonov variant configurations	144
5.8	Inverse PINN variant configurations	145
5.9	Six-variant benchmark results	147
5.10	Tikhonov v3 vs v4 settings	149
5.11	PINN v3 vs v4 settings	150
5.12	v4 benchmark results	152

Introduction

0.1 Motivation

Most introductory treatments of quantum mechanics rely on exactly solvable systems such as the infinite square well, the hydrogen atom, and the harmonic oscillator. These examples are pedagogically convenient because the Schrödinger equation admits closed-form solutions [1, 2], but they are the exception rather than the rule. The overwhelming majority of physically relevant quantum systems have no analytical solution, and progress in these fields depends on numerical methods capable of approximating the wave function, its time evolution, and the potential that generates it.

Two distinct classes of problems are found here. The forward problem is: given a potential $V(x)$ and an initial wave function $\psi(x, 0)$, how does the system evolve in time? This is the more familiar of the two, and decades of numerical analysis have produced reliable solvers like finite-difference methods, the Crank–Nicolson scheme, and explicit fast Fourier approaches such as Visscher’s method, each with well-understood stability and convergence properties [3, 4, 5]. The inverse problem is far less forgiving. Given observations of the wave function or its evolution, what potential produced them? Unlike the forward problem, this is typically ill-posed in the sense of Hadamard, where small errors in the observed data can produce large and physically unrealistic fluctuations in the recovered potential [6, 7]. Tikhonov regularization and related techniques exist precisely to stabilize this inversion [8, 9],

but they require careful tuning of regularization parameters and remain sensitive to noise and data sparsity.

Although the forward problem has been extensively studied, inverse quantum problems remain comparatively less explored, particularly in a unified framework that allows direct comparison with modern data-driven approaches. This gap is especially relevant when considering whether new machine learning-based methods genuinely improve upon classical numerical techniques or merely reproduce them in a different form.

Over the last several years, a different approach has emerged from the machine learning community. Although the idea of using neural networks to solve differential equations dates back to the work of Lagaris et al. [10], the formulation introduced by Raissi et al. [11] gave rise to physics-informed neural networks (PINNs), which embed the governing differential equation directly into the loss function used to train a neural network. The network is penalized not only for mismatching available data, but also for violating the underlying physical laws. This provides a unified framework that can, in principle, address both forward and inverse problems without requiring different numerical schemes for each [12]. For the inverse problem in particular, this is an appealing property, since the regularization imposed explicitly in classical approaches can instead emerge implicitly through the physical constraints embedded in the learning process.

Figure 1 shows the rapid expansion of the PINN literature between 2018 and 2025. What began as a methodological curiosity has evolved into a rapidly growing research field. Figure 2 breaks this corpus down by scientific field. Note that field categories are non-exclusive: a single publication indexed under several fields (e.g., both computer science and engineering) is counted once in each, so the field totals in Figure 2 exceed the number of unique publications shown in Figure 1.

Figure 2 highlights the disciplinary distribution of this corpus. Engineering, computer science, and mathematics dominate, while physics and astronomy remain

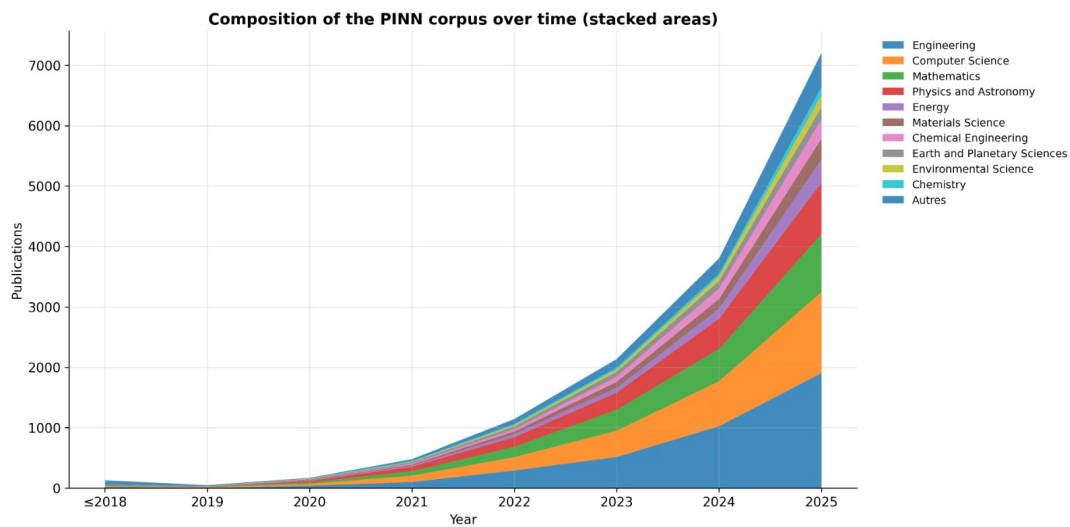


Figure 1: Composition of the body of publications devoted to PINNs over time (2018–2025), broken down by scientific field. Data for 2025 is partial.

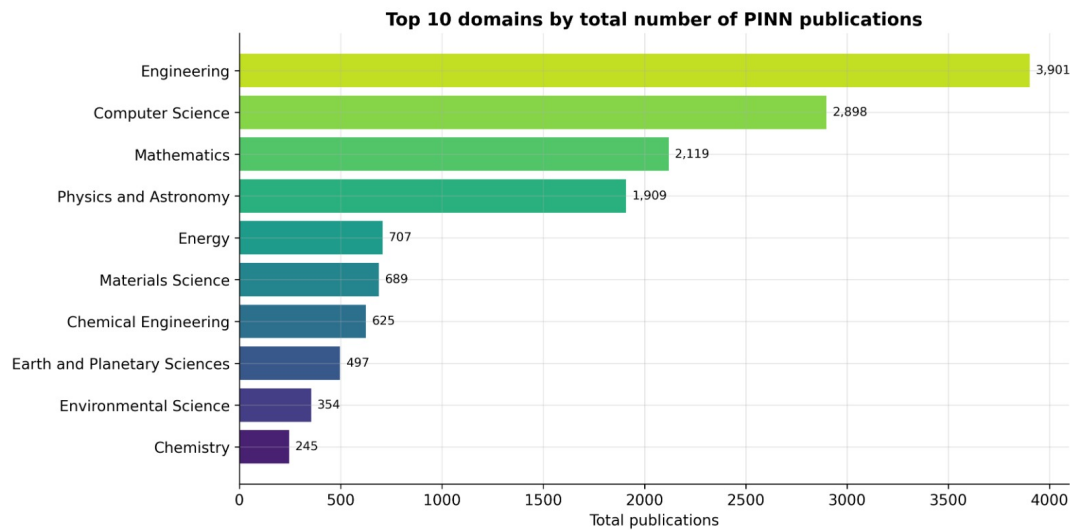


Figure 2: Top 10 scientific fields by total number of PINN publications (2018–2025). A publication may be indexed under multiple fields, so the totals exceed the corpus size.

significant but not leading. This distribution reveals an important imbalance: although PINNs were originally motivated by physical systems, their adoption has been far more intensive in engineering and data-driven sciences than in quantum physics itself. This observation motivates a systematic evaluation of their relevance in genuinely quantum settings, where analytical ground truth solutions are available for benchmarking.

This thesis asks whether that promise holds for a controlled, well-understood quantum system. The one-dimensional quantum harmonic oscillator is chosen as a benchmark rather than as a source of novel physical results [13]. Because its eigenstates, eigenvalues, and time evolution are known in closed form, any discrepancy between numerical or learning-based methods and the exact solution can be attributed to the method itself rather than uncertainty in the physics. This makes it possible to evaluate PINNs and classical solvers on equal footing, isolating their strengths and limitations.

0.2 Problem Statement

The work is centered on the time-dependent Schrödinger equation for the one-dimensional quantum harmonic oscillator,

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

Two complementary tasks are addressed.

0.2.1 Forward Task

The forward task considers the case where the potential $V(x)$, the mass m , and the initial condition $\psi(x, 0)$ are known. The goal is to predict the wave function $\psi(x, t)$ for $t > 0$.

A physics-informed neural network (PINN) is trained for this forward problem and its results are compared against a Crank–Nicolson finite difference scheme, which serves as a reliable numerical reference solution.

0.2.2 Inverse Task

The inverse task consists of reconstructing the potential $V(x)$ from observed samples of the wave function $\psi(x, t)$, which may be exact or corrupted with noise to simulate realistic experimental conditions.

In this work, a PINN-based inversion method is compared against classical Tikhonov regularization, which is a standard approach for stabilizing ill-posed inverse problems.

0.3 Objectives of the Thesis

- Review classical numerical methods used as benchmarks: finite difference time-stepping for the forward problem, and Tikhonov regularization for the inverse problem.
- Build and validate a forward PINN against the Crank–Nicolson solution.
- Build and validate an inverse PINN against Tikhonov-regularized inversion.
- Compare classical and physics-informed methods in terms of accuracy, computational cost, and robustness to noise.
- Assess under what conditions PINNs can be a credible alternative in quantum mechanical problems.

0.4 Thesis Structure

The remainder of this thesis is organized as follows.

Chapter 1 introduces the theoretical framework of quantum mechanics and the harmonic oscillator.

Chapter 2 presents classical numerical methods for both forward and inverse problems.

Chapter 3 introduces physics-informed neural networks and their mathematical formulation.

Chapter 4 details the forward PINN and compares it with Crank–Nicolson.

Chapter 5 presents the inverse PINN and compares it with Tikhonov regularization.

Chapter 6 concludes the thesis and discusses limitations and perspectives.

Chapter 1

The Schrödinger Equation: Mathematical Structure and Problem Formulation

The Schrödinger equation has an important position in the domain of partial differential equations. It is linear, but its solutions describe probabilities, not definite outcomes. It evolves smoothly in time but a single measurement collapses everything it describes. While it governs systems at the smallest physical scales, it uses the same mathematical structures, from eigenvalue problems to Hilbert spaces, and operator theory that appear in modern computational mathematics. This chapter builds the mathematical foundation that the rest of the thesis rests on. It begins with the physical origins of the Schrödinger equation as a PDE, establishes the operator framework through which quantum observables are defined, and develops the distinction between forward and inverse problems that motivates the use of both classical numerical solvers as well as modern solvers, such as physics-informed neural networks [11]. The goal is to give a precise account of the mathematical structure that a computational method must respect in order to produce physically meaningful results.

1.1 From Determinism to Wave Functions

In classical mechanics, the motion of particles is described using deterministic quantities such as position, velocity, and momentum, governed by differential equations that extend naturally to partial differential equations when fields and spatial distributions are involved. At microscopic scales, physical interactions are described by energy conservation, which defines an energy landscape represented by the potential energy. In quantum mechanics, the state of a system is described by a wave function $\psi(x, t)$, obtained by solving the Schrödinger equation. Analogous to Newtonian equations, given specific initial and boundary conditions, this equation determines the evolution of the system in time. However, the equation itself does not specify a single deterministic trajectory, but rather describes all possible allowed states of the system.

The Schrödinger equation therefore provides the fundamental partial differential equation describing the evolution of quantum systems. The wave function becomes the unknown function defined over space and time whose behavior is determined by the structure of the PDE together with the imposed initial and boundary conditions.

1.2 Probabilistic Nature and Wave Function Interpretation

The physical interpretation of the wave function is given by the Born statistical interpretation [14]. The squared magnitude of the wave function represents a probability density: $|\psi(x, t)|^2$. The probability of finding a particle between positions a and b at time t is given by

$$P(a \leq x \leq b) = \int_a^b |\psi(x, t)|^2 dx \quad (1.1)$$

Quantum mechanics does not predict exact measurement outcomes but instead

provides probabilities of possible measurement results [1].

This probabilistic description of the wave function is related to how quantum measurements are understood. Quantum systems exhibit two types of evolution. The first is the smooth time evolution governed by the Schrödinger equation. The second is the measurement process, where the wave function appears to collapse to a localized state corresponding to the measured value. Another distinction concerns the nature of measurable quantities. Discrete variables are those that take on isolated, specific values, such as the integer indices n labeling bound states in an infinite square well or a harmonic oscillator, where probabilities $P(j)$ are defined such that they satisfy the normalization condition:

$$\sum_j P(j) = 1.$$

Continuous variables like position x or momentum p fill out an entire range, requiring a probability density $\rho(x)$ where the probability of a measurement falling within an infinitesimal interval dx is $\rho(x)dx$, governed by the integral normalization:

$$\int \rho(x) dx = 1.$$

Physical states must satisfy the probabilistic interpretation of the wave function. The total probability of finding the particle anywhere in space must equal one:

$$\int_{-\infty}^{+\infty} |\psi(x, t)|^2 dx = 1 \tag{1.2}$$

Acceptable states must therefore be square integrable so that normalization is possible, and mathematically admissible wave functions belong to the Hilbert space $L^2(\mathbb{R})$. For a real potential $V(x)$, the Schrödinger equation implies a continuity equation for the probability density $\rho(x, t) = |\psi(x, t)|^2$:

$$\frac{\partial \rho}{\partial t} + \frac{\partial j}{\partial x} = 0, \quad j(x, t) = \frac{\hbar}{2mi} \left(\psi^* \frac{\partial \psi}{\partial x} - \psi \frac{\partial \psi^*}{\partial x} \right) \quad (1.3)$$

where $j(x, t)$ is the probability current. Integrating over all space, and since $\psi \rightarrow 0$ as $|x| \rightarrow \infty$ for square-integrable states, the current vanishes at infinity and the total probability is conserved over time:

$$\frac{d}{dt} \int_{-\infty}^{+\infty} |\psi(x, t)|^2 dx = 0 \quad (1.4)$$

meaning probability is neither created nor destroyed but redistributed in space.

An important feature of the wave function is its complex mathematical structure, which allows it to describe both amplitude and phase information. The amplitude determines probability density and the phase is responsible for interference effects. A general wave function can be written in polar form:

$$\psi(x, t) = R(x, t)e^{i\theta(x, t)},$$

and the complex exponential form allows oscillatory behavior

$$\psi(x, t) \sim e^{i(kx - \omega t)}$$

rather than purely exponential growth or decay. The imaginary unit is essential for oscillatory time evolution and conservation of probability.

1.3 Operators and Physical Observables

Physical observables such as position, momentum, and energy are represented by operators acting on the wave function. The wave function represents the state of the system, while measurements correspond to extracting physical information from this state using operators.

For an observable represented by an operator A , the measurement is described by the eigenvalue equation:

$$A\psi = a\psi \quad (1.5)$$

where a is the measurable eigenvalue associated with the eigenstate ψ .

In order for measurable quantities to be real-valued, observables must be represented by Hermitian operators. An operator is Hermitian if:

$$\langle \phi | A\psi \rangle = \langle A\phi | \psi \rangle \quad (1.6)$$

for all ϕ, ψ . Hermitian operators guarantee real eigenvalues, which correspond to possible measurement outcomes. When an observable is measured, the system collapses into one of the eigenstates of the corresponding operator, and the measured value is the associated eigenvalue.

For a system in a normalized state ψ , repeated measurements of an observable A yield on average the expectation value:

$$\langle A \rangle = \langle \psi | A | \psi \rangle \quad (1.7)$$

The statistical spread of the measurement outcomes around this average is quantified by the standard deviation:

$$\sigma_A = \sqrt{\langle A^2 \rangle - \langle A \rangle^2} \quad (1.8)$$

These algebraic properties also determine which quantities can be measured simultaneously, a structure formalized through the commutator of the corresponding operators. Some operators do not commute. The commutator between two operators A and B is defined as:

$$[A, B] = AB - BA \quad (1.9)$$

If two operators commute, their observables can be measured simultaneously with precision. If they do not commute, simultaneous precise measurement is impossible. More generally, the uncertainty relation reads [15]:

$$\sigma_A \sigma_B \geq \frac{1}{2} |\langle [A, B] \rangle| \quad (1.10)$$

For position and momentum, the canonical commutation relation is:

$$[x, p] = i\hbar \quad (1.11)$$

This leads to the Heisenberg uncertainty principle, expressed as [16]:

$$\Delta x \cdot \Delta p \geq \frac{\hbar}{2} \quad (1.12)$$

This limitation is not due to experimental imperfections but arises from the wave nature of quantum states. Gaussian wave packets represent minimum-uncertainty states that saturate the bound.

The energy-time uncertainty relation has a different interpretation because time is not a dynamical operator in quantum mechanics. It is expressed as [16]:

$$\Delta E \cdot \Delta t \geq \frac{\hbar}{2} \quad (1.13)$$

Here, Δt represents the characteristic time scale over which a system changes, rather than measurement uncertainty in time itself. This relation connects energy uncertainty to the lifetime of quantum states.

1.4 Time-Dependent Schrödinger Equation

Let $\Omega \subset \mathbb{R}^d$ and $t \in [0, T]$. The wave function is defined as:

$$\psi : \Omega \times [0, T] \rightarrow \mathbb{C}.$$

For each time t , the physical state satisfies $\psi \in L^2(\Omega)$, with the normalization condition (1.2). Thus, the Schrödinger equation is posed for complex-valued functions in the Hilbert space $L^2(\Omega)$ [17].

As a partial differential equation, the time-dependent Schrödinger equation (TDSE) is written as:

$$i\hbar \frac{\partial \psi}{\partial t} = H\psi \tag{1.14}$$

where, for a single particle in a potential $V(x)$, the Hamiltonian is defined as:

$$H = -\frac{\hbar^2}{2m} \nabla^2 + V(x) \tag{1.15}$$

The TDSE then takes the explicit form:

$$i\hbar \frac{\partial \psi}{\partial t} = -\frac{\hbar^2}{2m} \nabla^2 \psi + V(x)\psi \tag{1.16}$$

The equation is first order in time and second order in space. It is linear in ψ , complex-valued, and dispersive rather than diffusive, reflecting the wave nature of quantum evolution and distinguishing it structurally from parabolic equations such as the heat equation.

The equation conserves the L^2 norm (1.4). This conservation law is essential for the stability and physical consistency of numerical methods used to simulate the equation.

1.5 Time-Independent Schrödinger Equation

If the Hamiltonian does not depend explicitly on time, solutions of the TDSE can be found using separation of variables. Assuming

$$\psi(x, t) = \phi(x)T(t),$$

the Schrödinger equation separates into spatial and temporal components:

$$i\hbar \frac{1}{T} \frac{dT}{dt} = \frac{1}{\phi} H\phi = E.$$

The separation constant is identified as the energy E , which leads to the temporal equation

$$T(t) = \exp\left(-\frac{iEt}{\hbar}\right)$$

and the spatial equation known as the Time-Independent Schrödinger Equation (TISE):

$$H\psi(x) = E\psi(x) \tag{1.17}$$

Using the explicit form of the Hamiltonian, this is expressed as:

$$-\frac{\hbar^2}{2m} \nabla^2 \psi + V(x)\psi = E\psi \tag{1.18}$$

The TISE is an eigenvalue problem. The allowed energies correspond to eigenvalues of the Hamiltonian operator, and the admissible eigenfunctions represent the corresponding spatial states. The problem is second order in space and elliptic for reasonable potentials.

Energy levels may be degenerate if multiple eigenstates correspond to the same eigenvalue. Degeneracy typically arises from symmetries of the physical system.

The time evolution of eigenstates is given by

$$\psi_n(x, t) = \phi_n(x) \exp\left(-\frac{iE_n t}{\hbar}\right),$$

which preserves the probability density since the time dependence appears only as a phase factor. General solutions can be expressed as superpositions of eigenstates [1]:

$$\psi(x, t) = \sum_n c_n \phi_n(x) \exp\left(-\frac{iE_n t}{\hbar}\right) \quad (1.19)$$

For time-independent Hamiltonians, the time evolution corresponds to phase rotation of the state vector in complex vector space. Each energy eigenstate acquires a time-dependent phase factor.

1.6 Boundary Conditions for the Schrödinger PDE

Boundary conditions act as physical and mathematical filters that restrict the set of acceptable solutions of the Schrödinger equation. The Schrödinger equation may admit many mathematical solutions, but only those satisfying physical boundary conditions correspond to realistic quantum states.

Since the Schrödinger equation contains a second-order spatial derivative, the wave function must generally be continuous. A discontinuity in probability density would imply an instantaneous jump in the probability of finding a particle between two nearby points, which is physically impossible. The first derivative of the wave function is also generally continuous, except in the presence of singular potentials such as infinite potential barriers.

Several distinct types of boundary conditions arise depending on the physical configuration of the system. Dirichlet boundary conditions fix the value of the wave function at the boundary of the domain: for a domain $[x_{\min}, x_{\max}]$, they take

the general form $\psi(x_{\min}, t) = f_1(t)$ and $\psi(x_{\max}, t) = f_2(t)$, where f_1 and f_2 are prescribed functions. The most common example is the infinite potential well, where the particle is confined to an interval $[0, a]$ and the wave function must satisfy $\psi(0, t) = 0$ and $\psi(a, t) = 0$ at all times. These constraints lead to standing-wave solutions and a discrete energy spectrum, turning the Schrödinger equation into an elliptic boundary value problem with quantized eigenvalues.

Neumann-type conditions describe continuity of derivatives rather than values of the wave function. When solving systems such as the finite square well, continuity requires that the slope of the wave function inside the potential matches the slope outside at the interface: $\psi'_{\text{in}}(a) = \psi'_{\text{out}}(a)$. This condition is only relaxed at points where the potential is infinite, such as at the walls of an infinite square well or at the site of a Dirac delta function [1].

For central potentials where V depends only on the radial distance r , the Schrödinger equation is solved in spherical coordinates, yielding a radial equation for $R(r)$, commonly simplified via the substitution $u(r) = rR(r)$. The most critical constraint is the behavior at the origin: since $R(r) = u(r)/r$, a nonzero value of $u(0)$ would cause $R(r)$ to diverge as $1/r$ at the origin. A general argument therefore requires $u(0) = 0$ for the state to constitute a valid solution of the Hamiltonian.

Bloch's theorem applies to particles moving in a periodic potential, such as an electron in a crystal lattice where $V(x) = V(x + a)$. It states that stationary states take the form $\psi(x) = \exp(iqx)u(x)$, where $u(x)$ is a periodic function with the same periodicity as the potential. This implies that the states are traveling waves modulated by the lattice, meaning an electron can propagate through a perfect crystal without scattering.

Scattering states correspond to solutions of the Schrödinger equation associated with the continuous spectrum of the Hamiltonian operator. Unlike bound states, scattering states are non-normalizable and are interpreted through wave packets constructed from superpositions of eigenstates. The asymptotic behavior of scatter-

ing states provides the appropriate boundary conditions for open quantum systems.

In one dimension, the wave function typically satisfies

$$\psi(x) \sim \begin{cases} A \exp(ikx) + B \exp(-ikx), & x \rightarrow -\infty \\ F \exp(ikx), & x \rightarrow +\infty \end{cases} \quad (1.20)$$

representing incident, reflected, and transmitted waves. In three dimensions, asymptotic solutions take the form [1]

$$\psi(r, \theta) \sim \exp(ikz) + \frac{f(\theta) \exp(ikr)}{r}, \quad (1.21)$$

where the $1/r$ decay ensures conservation of probability at large distances.

Singular interactions in quantum mechanics are often modeled using distributional potentials, the most common being the Dirac delta function, defined by

$$\int f(x) \delta(x - a) dx = f(a).$$

For a delta potential $V(x) = \alpha \delta(x - x_0)$, integrating the TISE across the singularity shows that the wave function remains continuous while its derivative exhibits a finite jump. The choice of boundary conditions thus plays a determining role in selecting the physically admissible solutions of the Schrödinger equation.

1.7 Self-Adjointness Properties

Boundary conditions not only select physically meaningful wave functions, but also define the domain of the Hamiltonian operator acting on these functions. This domain specification is essential: only with the correct domain does the Hamiltonian become self-adjoint.

An operator is self-adjoint ($Q = Q^\dagger$) if it is identical to its adjoint in both its mathematical action and its domain, which is the set of vectors upon which it

acts. Self-adjointness is required for an operator to represent a physical observable, because it ensures that eigenvalues are real and eigenfunctions are complete [17].

In finite-dimensional spaces, the domain is the entire space and the distinction is negligible, but in infinite-dimensional Hilbert spaces, boundary conditions are critical; for example, the momentum operator on a finite interval may be Hermitian but fail to be self-adjoint if the domain of the operator and its adjoint do not match exactly [17]. Self-adjointness of the discrete Hamiltonian is therefore not merely a mathematical formality but a requirement for the physical consistency of any numerical scheme.

1.8 Examples of Quantum Systems

Three canonical quantum systems illustrate how boundary conditions, operator structure, and energy quantization arise in practice.

The infinite square well describes a particle confined between $x = 0$ and $x = a$ by infinite potential walls. The TISE takes the form

$$-\frac{\hbar^2}{2m} \frac{d^2\psi}{dx^2} = E\psi,$$

subject to the boundary conditions $\psi(0) = \psi(a) = 0$. These constraints lead [1] [15] to quantized energies:

$$E_n = \frac{n^2\pi^2\hbar^2}{2ma^2} \tag{1.22}$$

and standing-wave eigenfunctions:

$$\psi_n(x) = \sqrt{\frac{2}{a}} \sin\left(\frac{n\pi x}{a}\right) \tag{1.23}$$

The finite square well retains the same second-order structure but replaces the infinite walls with finite potential barriers, allowing the system to admit both discrete

bound states and continuous scattering states. The wave function and its derivative must match at the interfaces, corresponding to the Neumann-type boundary conditions discussed earlier.

The quantum harmonic oscillator represents a smooth confining potential that grows indefinitely at large distances:

$$V(x) = \frac{1}{2}m\omega^2x^2 \quad (1.24)$$

Because the potential grows as $|x| \rightarrow \infty$, the wave function must decay to zero at infinity in order to remain normalizable. In contrast to systems with hard boundaries such as infinite potential wells, confinement arises from the smooth shape of the potential itself rather than from explicit boundary walls. Solving the TISE

$$-\frac{\hbar^2}{2m} \frac{d^2\psi}{dx^2} + \frac{1}{2}m\omega^2x^2\psi = E\psi$$

yields a discrete set of energy eigenvalues [1]:

$$E_n = \hbar\omega \left(n + \frac{1}{2} \right) \quad (1.25)$$

and eigenfunctions expressed as Hermite polynomials multiplied by a Gaussian factor. This system demonstrates the properties of self-adjoint operators on an unbounded domain with decay conditions at infinity. The harmonic oscillator is of particular relevance to the present work, as it constitutes the benchmark system against which both classical numerical methods and physics-informed neural networks will be evaluated in the chapters that follow.

1.9 Variants of the Schrödinger Equation

The standard Schrödinger equation describes non-relativistic quantum systems and can be formulated as a linear partial differential equation governing the evolution

of a complex wave function. Several related equations arise in different physical regimes and lead to PDEs with distinct mathematical structures.

The Dirac equation provides a relativistic description of quantum particles and takes the form of a system of first-order linear PDEs:

$$(i\gamma^\mu \partial_\mu - m)\psi = 0.$$

Unlike the Schrödinger equation, which is second order in space, the Dirac equation is first order in both time and space and describes spin- $\frac{1}{2}$ particles.

The Klein–Gordon equation describes scalar particles and has the structure of a second-order hyperbolic PDE:

$$(\square + m^2)\psi = 0.$$

The Pauli equation extends the Schrödinger equation to include spin effects and interactions with electromagnetic fields:

$$i\hbar \frac{\partial \psi}{\partial t} = \left[\frac{1}{2m}(\mathbf{p} - q\mathbf{A})^2 + q\phi - \boldsymbol{\mu} \cdot \mathbf{B} \cdot \boldsymbol{\sigma} \right] \psi.$$

The present work is restricted to the non-relativistic single-particle regime, for which the standard Schrödinger equation provides the appropriate governing equation.

1.10 Forward and Inverse Problems in Quantum Mechanics

The primary task of physicists has traditionally been to predict the behavior of a quantum system from known physical laws and system parameters. The starting point consists of well-defined quantities such as the system’s potential energy, the

initial wave function, and the boundary conditions imposed.

Given this information, the goal is to determine how the wave function evolves in space and time, and how physical observables behave. Mathematically, this corresponds to a forward (or direct) quantum problem, where the unknown field, which is the wave function $\psi(x, t)$, is computed by solving the time-dependent Schrödinger equation with the initial state $\psi(x, 0)$ and appropriate boundary conditions. Solving this forward problem yields predictions of energy spectra, probability distributions, and time evolution directly from the PDE structure.

While forward problems have been the mainstay of quantum physics, modern computational methods increasingly address inverse questions: instead of computing the wave function from a known potential, the objective becomes inferring unknown potentials or interaction parameters from observed data. These inverse quantum problems are more challenging, as they require reconstructing the PDE inputs from partial or noisy measurements of the system's response.

The study of quantum systems thus naturally divides into forward problems, which simulate the system given the PDE and its inputs, and inverse problems, which seek to reconstruct the PDE inputs from observed outputs. Understanding both perspectives underpins modern applications, especially in data-driven physics approaches such as physics-informed neural networks (PINNs) [11].

1.11 Forward (Direct) Problems

A forward problem is an analysis oriented along a cause–effect sequence, where the objective is to determine the output behavior of a physical system given its governing physical laws, inputs, and parameters. In quantum mechanics, this corresponds to predicting the evolution of quantum states from known physical configurations.

The mathematical formulation of forward problems is expressed using functional

analytic and operator-theoretic terms. Let X denote the space of input quantities such as potentials or initial wave fields, Y the space of observable outputs such as wave functions, P the parameter space of physical system parameters, and $A(p)$ an operator mapping $X \rightarrow Y$ associated with parameter $p \in P$. The forward problem can then be formulated as $g = A(p)f$, where $f \in X$ represents the system input and $g \in Y$ represents the system output [7]. Computing the output requires solving ordinary differential equations, partial differential equations, or integral equations.

Although forward problems are considered mathematically well-posed, their numerical computation can still be challenging due to the complexity of the underlying PDEs. For rigorous mathematical analysis, these problems are typically defined on complete Hilbert spaces, allowing the use of spectral methods and singular value decompositions for operator analysis. Continuity of the operator A ensures stability, meaning that small perturbations in the input space produce small variations in the output space.

In quantum mechanics, forward problems correspond to predicting the time evolution of quantum states from known physical information. Given a potential energy function $V(x)$, an initial wave function $\psi(x, 0)$, and appropriate boundary conditions, the forward problem consists of solving the Schrödinger equation (1.16) to obtain the wave function evolution $\psi(x, t)$. This formulation is an initial-boundary value PDE problem defined in Hilbert space $L^2(\Omega)$, where the evolution is unitary and probability is conserved.

Forward quantum PDE problems are generally characterized by four properties. First, with respect to well-posedness, direct problems satisfy Hadamard's criteria, meaning a solution exists, is unique, and depends continuously on the data [18]. Small variations in the input data therefore produce only small variations in the output solution. Second, forward problems are associated with a loss of information, in the sense that the solution is generally much smoother than the input data; for instance, a scattered wave produced by a rough obstacle remains smooth. Third,

because they follow a cause-effect sequence, forward problems represent a transition from a physical quantity with a certain information content to another quantity with smaller information content. Fourth, despite their well-posed character, forward quantum simulations often require dedicated numerical methods. The most widely used approaches include finite difference methods, which discretize the spatial and temporal derivatives directly; spectral methods, which expand the solution in terms of basis functions; and finite element methods, which decompose the domain into elements and solve the equation locally over each [19].

1.12 Inverse Problems

Inverse problems represent a fundamental shift in the direction of analysis relative to direct problems. While direct problems use known inputs, system parameters, and boundary conditions to predict the output via the forward operator $A(p)$, inverse problems aim to reconstruct unknown causes from measured effects. In quantum mechanics, this corresponds to using observed wave functions or scattering data to determine unknown potentials, system parameters, or initial states.

Inverse problems are defined as problems of determining the internal structure or past state of a system from indirect measurements [7]. Two problems are considered inverse to each other if the formulation of each requires full or partial knowledge of the other. The problem studied first is labeled direct, while the secondary, often more difficult problem, is labeled inverse. The mathematical description is framed using the same functional analytic terms introduced for direct problems, namely the input space X , the output space Y , the parameter space P , and the forward operator $A(p)$ mapping $X \rightarrow Y$ for $p \in P$.

Based on these spaces, inverse problems are categorized into two types. In a reconstruction problem, the output $g \in Y$ and parameters $p \in P$ are known, and the objective is to recover the input $f \in X$ such that

$$g = A(p)f \tag{1.26}$$

In an identification problem, both the input $f \in X$ and the output $g \in Y$ are known, and the goal is to determine the system parameters $p \in P$ such that $A(p)f = g$.

Unlike direct problems, which are generally well-posed, inverse problems are almost always ill-posed. According to Hadamard's criteria, a problem is ill-posed if it fails any of three requirements [18]. The first is existence: a solution must exist for arbitrary data, but in inverse problems, noisy measurements may fall outside the range of the operator A , meaning no theoretical solution exists that perfectly matches the data. The second is uniqueness: there must be only one solution, but inverse problems often suffer from non-uniqueness, where different inputs produce identical outputs, making it impossible to distinguish between them without additional information. The third is stability: the solution must depend continuously on the data, but inverse problems are highly sensitive to measurement errors, and a small perturbation in the data can produce a drastically different and physically meaningless solution.

To stabilize the solution, approximate formulations with regularization are employed. Tikhonov regularization, for instance, addresses instability by minimizing the functional [8] [20]

$$J(x) = \|Kx - y^\delta\|^2 + \alpha\|x\|^2 \tag{1.27}$$

where x is the unknown quantity to reconstruct, K is the forward operator mapping x to predicted measurements, y^δ is the measured data including noise, $\|\cdot\|$ denotes the norm in the appropriate function space, and α is the regularization parameter controlling the trade-off between data fidelity and solution smoothness. This formulation ensures that the reconstructed solution remains stable and physi-

cally meaningful even when the measured data contains noise.

Numerical methods for solving inverse problems focus on overcoming ill-posedness, where measurement noise would otherwise be amplified into physically meaningless solutions. The guiding principle for these methods is to seek approximate solutions satisfying additional *a priori* constraints based on physical properties such as smoothness or prescribed bounds. Regularization techniques such as Tikhonov and truncated singular value decomposition control noise amplification, while optimization and iterative methods based on least-squares formulations and linearization make repeated use of forward problem evaluations [7] [21] [22]. Computational frameworks incorporating finite element methods, Monte Carlo sampling, and Bayesian priors implement these reconstructions in practice, and parameter selection strategies such as the L-curve method and cross-validation [21] are used to balance accuracy and stability.

1.13 Summary

The mathematical structure established in this chapter shows why solving quantum PDEs computationally is a challenging task. Forward problems, despite being well-posed, involve complex-valued dispersive equations posed on Hilbert spaces, where conservation of the L^2 norm must be respected at every step. Inverse problems make this even harder, the loss of information in the forward direction means that reconstructing potentials or initial states from measured data is ill-posed by nature, sensitive to noise, and in need of regularization to produce stable results. Classical numerical methods address these challenges through discretization schemes and iterative solvers, and understanding their strengths and limitations is essential before introducing a learning-based alternative. Chapter 2 examines these methods in detail, covering the established numerical approaches for both forward and inverse Schrödinger problems that physics-informed neural networks will later be measured

against.

Chapter 2

Classical Numerical Solvers for Forward and Inverse Schrödinger Equation

The mathematical foundations of quantum mechanics, presented in Chapter 1, provide the theoretical framework for describing quantum systems, but obtaining solutions to the Schrödinger equation in practice requires numerical methods. Analytical solutions exist only for a handful of idealized systems like the infinite square well, the hydrogen atom, and the harmonic oscillator [1], and even these become intractable the moment realistic interactions or multi-particle effects are introduced.

This chapter presents the classical numerical approach to both the forward and inverse Schrödinger problems, with a focus on the Finite Difference Method (FDM) as the foundational discretization framework. It begins by building the method from first principles, starting from Taylor series approximations of derivatives [4], constructing the discrete Hamiltonian matrix [13], and developing the spatial discretization of the Schrödinger equation.

The key concepts governing numerical stability and convergence are then examined, including the local truncation error, the amplification factor, the Lax equiv-

alence theorem, and von Neumann stability analysis, before presenting the main time-stepping schemes including both explicit and implicit, with a particular emphasis on the Crank–Nicolson method [3], which stands as the standard classical solver for the time-dependent Schrödinger equation.

The second part of the chapter addresses the classical inverse problem including recovering an unknown potential $V(x)$ from noisy observations of the wave function. The discrete forward operator is formulated, the ill-posedness of the inverse problem in the sense of Hadamard is established [18, 6], and Tikhonov regularization is introduced as the standard stabilization strategy [8].

The regularized functional, its gradient decomposed into a residual term and a penalty term, the role of the Jacobian in the optimization process, and the Morozov discrepancy principle for parameter selection [23] are all developed in detail.

The limitations of these classical methods such as computational cost, dimensional scaling, sensitivity to noise, and the absence of a unified forward–inverse framework all motivate the transition to Physics-Informed Neural Networks developed in Chapter 3.

2.1 Classical Forward Solver: Finite Difference Method

2.1.1 Core Idea

Differential equations describe how things change; they tell how a function behaves through its derivatives. In classical mechanics, the derivative of position gives velocity. In quantum mechanics, the derivatives of the wave function in space and time encode how the quantum state evolves. But differential equations live in a continuous world, meaning infinitely many points and infinitely many values, while computers only handle finite data, such as numbers at specific discrete locations.

Since we cannot directly evaluate continuous derivatives on a machine, we need a practical approximation strategy.

The Finite Difference Method gives a simple answer, rooted in the very definition of the derivative from calculus:

$$u'(x) = \lim_{h \rightarrow 0} \frac{u(x+h) - u(x)}{h}$$

[4]

The derivative is just the slope between two nearby points as the gap between them shrinks to zero. FDM says to not take the limit. We keep h small but finite, and we pick a discrete set of points on the function, approximating the slope using actual computable values:

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

A derivative, which is a concept from continuous mathematics, is replaced by simple subtraction and division. The smooth curve becomes a collection of points, and we estimate its slope by walking between a few measured points instead of knowing the exact shape everywhere.

Once any derivative can be approximated by arithmetic, an entire differential equation can be treated the same way. Every derivative in the equation is replaced by its finite difference version, evaluated on a grid of discrete points. What was an infinite-dimensional problem in continuous space becomes a large but finite system of algebraic equations, and difficult calculus turns into linear algebra, which a computer solves straightforwardly as a matrix problem [4].

In the Schrödinger case, this is particularly powerful. The Schrödinger equation is a partial differential equation involving derivatives in both space and time, and analytically it yields clean solutions only for a handful of idealized systems such as the harmonic oscillator, the hydrogen atom, and the infinite square well. For

anything more realistic, a numerical approach is required [24]. FDM handles this by discretizing the spatial domain into a grid of N points separated by Δr , and replacing the spatial second derivative (the Laplacian) with a three-point centered formula. Time is discretized separately with its own step Δt . The continuous wave function $\psi(x, t)$, which was a smooth function over all space and time, becomes a vector of numbers updated step by step. The quantum evolution is carried entirely by finite arithmetic differences [13].

The consequence of this discretization is that the wave function becomes a vector of values at grid points, and the Hamiltonian operator, which in continuous quantum mechanics acts on wave functions through differential operators, becomes a matrix acting on that vector. The physics of the Schrödinger equation is preserved, but expressed entirely in the language of linear algebra. This is the core idea of FDM: discretizing space and derivatives, and transforming a continuous physics problem into a finite matrix problem that a computer can solve.

2.1.2 Mathematical Foundations

Derivative Approximations from Taylor Series

Instead of working with a continuous function $u(x)$, we work with its values at a discrete set of points

$$r_1, r_2, \dots, r_N,$$

and reconstruct derivatives using only these values. The tool that makes this possible, and that underlies every finite difference formula, is the Taylor series.

The Taylor Series: The Engine Behind Everything Before any finite difference formula can be derived or understood, we need to know where it comes from. The Taylor series states that any smooth function can be written as an infinite sum

around a point x [4]:

$$f(x+h) = f(x) + hf'(x) + \frac{1}{2}h^2f''(x) + \frac{1}{6}h^3f'''(x) + \dots \quad (2.1)$$

This is not an approximation, but an exact equality for smooth functions. The principle is that if we know a function at one point and all its derivatives there, we can know its value everywhere nearby. The h^n terms become smaller and smaller as $h \rightarrow 0$, so the further out we go in the series, the less those terms contribute. Every finite difference formula is obtained by rearranging this series to isolate the derivative we want.

The Forward Difference Formula Rearranging equation (2.1) and solving for $f'(x)$:

$$f'(x) = \frac{f(x+h) - f(x)}{h} - \frac{1}{2}hf''(x) - \dots$$

The first term is the forward difference formula. The remaining terms are the error, which is what we throw away when we use only two points. Written compactly using big- $\mathcal{O}$ notation:

$$f'(x) = \frac{f(x+h) - f(x)}{h} + \mathcal{O}(h) \quad (2.2)$$

[4][25]

The $\mathcal{O}(h)$ notation means the dominant error term is proportional to h : halve the step size, and halve the error. This is first-order accuracy. Despite its simplicity, however, the formula is geometrically biased because it only looks forward, ignoring everything behind the current point.

The Backward Difference Formula Expanding $f(x-h)$ instead:

$$f(x-h) = f(x) - hf'(x) + \frac{1}{2}h^2f''(x) - \dots$$

Solving for $f'(x)$:

$$f'(x) = \frac{f(x) - f(x-h)}{h} + \mathcal{O}(h) \quad (2.3)$$

[4, 25]

This has the same first-order accuracy and the same error magnitude, but now looks backwards. It carries an equal and opposite bias to the forward formula: it ignores where we are heading.

The Central Difference Formula: Error Cancellation Taking both Taylor expansions simultaneously, equation (2.1) for $f(x+h)$ and the analogous expansion for $f(x-h)$, and subtracting one from the other:

$$f(x+h) - f(x-h) = 2hf'(x) + \frac{1}{3}h^3f'''(x) + \dots$$

Solving for $f'(x)$ [4, 25]:

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h} + \mathcal{O}(h^2) \quad (2.4)$$

The h^2 terms in both expansions are identical and cancel exactly when we subtract. What remains is a much smaller error proportional to h^2 , which is second-order accuracy. Halve the step size, and the error drops by a factor of four. The improvement is a consequence of the symmetry of looking both left and right simultaneously. The forward bias and backward bias are mirror images of each other and therefore cancel out.

The Second Derivative Formula Rather than subtracting the two Taylor expansions, we now add them:

$$f(x+h) + f(x-h) = 2f(x) + h^2f''(x) + \frac{1}{12}h^4f^{(4)}(x) + \dots \quad (2.5)$$

Solving for $f''(x)$ [4, 25]:

$$f''(x) = \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} + \mathcal{O}(h^2) \quad (2.6)$$

By symmetry, the odd-power error terms cancel exactly, giving second-order accuracy. This is the standard centered approximation to the second derivative, and it is of central importance to the Schrödinger equation since the kinetic energy operator contains precisely this term:

$$-\frac{\hbar^2}{2m} \frac{\partial^2 \psi}{\partial x^2}$$

The formula uses three points: the point itself and its two immediate neighbors. The $-2f(x)$ term in the numerator plays the role of a reference. It measures how much the function curves away from its central value by checking whether the middle point sits above or below the average of its two neighbors. If the middle point is lower than the average, the function curves upward, corresponding to positive curvature. If it sits higher, the function curves downward, corresponding to negative curvature. This represents the physical content of the kinetic energy term which measures the spatial curvature of the wave function.

Order of Accuracy Higher-order approximations can be obtained by combining values at more neighboring points. By including $f(x+2h)$, $f(x-2h)$, and so on, more error terms can be cancelled, achieving third- or fourth-order accuracy [25]. The general principle is that more points in the stencil means more cancellation and higher accuracy, at the cost of wider stencils and more complex boundary treatment.

To summarize the orders encountered so far:

Formula	Error	Order
Forward difference	$\mathcal{O}(h)$	First order
Backward difference	$\mathcal{O}(h)$	First order
Central difference	$\mathcal{O}(h^2)$	Second order
Higher-order stencils	$\mathcal{O}(h^3), \mathcal{O}(h^4)$	Third, fourth order

In this thesis, we focus on the second-order centered formula for the second derivative, given by equation (2.6), since it provides the right balance between accuracy and computational simplicity for the spatial discretization of the Schrödinger equation.

From Operators to Matrices

The Wave Function as a Vector In continuous quantum mechanics, the wave function $\psi(x, t)$ is defined at every point in space. Once we place a grid of N points separated by Δr on the spatial domain, we no longer track ψ everywhere. Instead, we store one complex number per grid point:

$$\boldsymbol{\psi} = \begin{pmatrix} \psi(r_1, t) \\ \psi(r_2, t) \\ \vdots \\ \psi(r_N, t) \end{pmatrix}$$

The wave function has become a vector. Once ψ is a vector, anything that acts on it must be a matrix. This is the entire reason operators become matrices in FDM.

The Kinetic Energy Matrix In the Schrödinger equation, the kinetic energy operator acts on ψ through the second spatial derivative:

$$\hat{T} = -\frac{\hbar^2}{2m} \frac{\partial^2}{\partial x^2}$$

We already know from equation (2.6) how to approximate this second derivative

at a grid point r_i :

$$\left. \frac{\partial^2 \psi}{\partial r^2} \right|_i \approx \frac{\psi_{i+1} - 2\psi_i + \psi_{i-1}}{\Delta r^2} \quad (2.7)$$

Writing this at every grid point simultaneously and collecting the coefficients of $\psi_1, \psi_2, \dots, \psi_N$ in each row, we obtain the discretized kinetic energy matrix [13]:

$$T = -\frac{\hbar^2}{2m\Delta r^2} \begin{pmatrix} -2 & 1 & 0 & \cdots \\ 1 & -2 & 1 & \cdots \\ 0 & 1 & -2 & \cdots \\ \vdots & & & \ddots \end{pmatrix} \quad (2.8)$$

This tridiagonal structure is not imposed artificially. It follows naturally from the second derivative formula. At each grid point, only the point itself and its two immediate neighbors appear. Every other entry is zero because distant points do not contribute to the local curvature. Each row is simply the coefficients of ψ_{i-1} , ψ_i , and ψ_{i+1} read directly from equation (2.7).

The Potential Energy Matrix The potential energy operator acts on ψ by multiplying it pointwise:

$$\hat{V}\psi(r_i) = V(r_i)\psi(r_i)$$

At each grid point, the potential simply multiplies the local value of the wave function by a scalar. There is no coupling to neighbors. In matrix form, this is a diagonal matrix [13]:

$$U = \begin{pmatrix} V(r_1) & 0 & \cdots \\ 0 & V(r_2) & \cdots \\ \vdots & & \ddots \end{pmatrix} \quad (2.9)$$

The diagonal structure reflects the local nature of the potential, it only acts at the point where it is evaluated, with no influence on neighboring points.

The Hamiltonian Matrix The Hamiltonian is the sum of kinetic and potential energy:

$$H = T + U \quad (2.10)$$

Adding equations (2.8) and (2.9) gives [13]:

$$H = \begin{pmatrix} \frac{\hbar^2}{m\Delta r^2} + V_1 & -\frac{\hbar^2}{2m\Delta r^2} & 0 & \cdots \\ -\frac{\hbar^2}{2m\Delta r^2} & \frac{\hbar^2}{m\Delta r^2} + V_2 & -\frac{\hbar^2}{2m\Delta r^2} & \cdots \\ 0 & -\frac{\hbar^2}{2m\Delta r^2} & \frac{\hbar^2}{m\Delta r^2} + V_3 & \cdots \\ \vdots & & & \ddots \end{pmatrix} \quad (2.11)$$

The structure of this matrix has a physical interpretation. The diagonal entries $\frac{\hbar^2}{m\Delta r^2} + V_i$ carry two contributions: how curved the wave function is at point i (kinetic energy) and the potential energy at that point. The off-diagonal entries $-\frac{\hbar^2}{2m\Delta r^2}$ show how strongly each point is coupled to its immediate neighbors through kinetic energy. The zeros appear naturally elsewhere: a point at position r_i has no direct interaction with a point at r_j when $|i - j| > 1$, because the second derivative only reaches one step in each direction.

The Hamiltonian is therefore the complete discrete rule governing the quantum system, the diagonal tells each point how it evolves locally, and the off-diagonals tell each point how it interacts with its surroundings. The Schrödinger equation now becomes a matrix equation acting on the vector ψ , ready for numerical time evolution.

Spatial Discretization of the Schrödinger Equation

After substituting the second derivative formula (2.7) into the Schrödinger equation at each grid point r_i , the continuous PDE becomes a system of N coupled ordinary differential equations in time. At grid point i :

$$i\hbar \frac{d\psi_i}{dt} = -\frac{\hbar^2}{2m\Delta r^2} (\psi_{i+1} - 2\psi_i + \psi_{i-1}) + V_i\psi_i \quad (2.12)$$

Written simultaneously for all N grid points in matrix form [13]:

$$i\hbar \frac{d\psi}{dt} = H\psi \quad (2.13)$$

where

$$\psi(t) = (\psi_1(t), \dots, \psi_N(t))^T$$

is the wave function vector and H is the discretized Hamiltonian matrix of equation (2.11).

Equation (2.12) is the fully discretized form of the Schrödinger equation. At each grid point, the time evolution of ψ_i depends only on its own value and the values at its two immediate neighbors ψ_{i-1} and ψ_{i+1} . Spatial derivatives have been completely replaced by finite differences. What remains is a system of ordinary differential equations in time, one for each grid point, all coupled through the Hamiltonian matrix H .

The question is now: how to advance this system forward in time, which is the subject of the following sections.

2.1.3 Key Concepts for Stability Analysis

Before presenting the time-stepping methods, we introduce the key concepts that will be used to evaluate them. These notions apply to all methods, explicit and implicit, and understanding them beforehand makes the analysis of each scheme more natural.

The Amplification Factor When any time-stepping scheme is applied, the wave function is multiplied by some factor at each step. The amplification factor g is that factor. It tells us: after one time step, by how much does each component of the wave function grow or shrink? In order to find it, we substitute a single Fourier mode $e^{i\xi x}$ into the difference equation. The scheme returns the same mode multiplied by

a complex number $g(\xi)$, that is the amplification factor for that wave number [4].

Probability Conservation and Unitarity As seen in Chapter 1, in quantum mechanics, total probability

$$\int |\psi|^2 dx$$

must always equal one [1, 15]. For this to remain true after every time step, the wave function cannot grow or shrink, it can only rotate in the complex plane. Mathematically it requires

$$|g| = 1$$

exactly. The three possible cases are:

Condition	Behaviour	Verdict
$ g > 1$	Wave grows, errors explode	Unstable
$ g < 1$	Wave shrinks, probability leaks	Stable but not unitary
$ g = 1$	Wave only rotates	Stable and unitary

Stability Condition The stability condition comes directly from demanding

$$|g| \leq 1.$$

We compute g for a given scheme and it comes out as an expression depending on Δt and Δx , and then ask: for what values of Δt does $|g|$ stay at or below one? The answer is the stability condition. It tells us how large the time step can be before errors start snowballing [4].

Local Truncation Error At every grid point, the finite difference formula is an approximation, not the exact derivative. The local truncation error (LTE) measures how big that gap is at a single point in a single step. It is estimated using Taylor series [4, 25]. For the centered second derivative formula used throughout this

chapter, the LTE is

$$\mathcal{O}(\Delta x^2),$$

meaning the gap shrinks quadratically as the grid is refined.

Consistency A scheme is consistent if its local truncation error vanishes as

$$\Delta t, \Delta x \rightarrow 0.$$

This means the discrete equations approximate the right continuous equation as the grid is refined [4], not some other equation entirely. Consistency is the minimum requirement for a scheme to make physical sense.

Convergence A scheme is convergent if the numerical solution approaches the true solution as the grid is refined:

$$\|U^n - u(t_n)\| \rightarrow 0 \quad \text{as } \Delta x, \Delta t \rightarrow 0 \quad (2.14)$$

This is the ultimate goal of any numerical method. Convergence is what we actually want but proving it requires both consistency and stability together [4].

The Lax Equivalence Theorem The Lax equivalence theorem ties everything together [4]:

$$\text{consistency} + \text{stability} \implies \text{convergence} \quad (2.15)$$

Consistency alone is not enough. The right equation can be approximated and still have errors explode if the scheme is unstable. Stability is the missing step that keeps accumulated local errors from contaminating the global solution.

Stability and Von Neumann Analysis A scheme is stable if small early mistakes do not snowball as the computation advances. Formally, the amplification

factor must satisfy

$$|g(\xi)| \leq 1$$

for all wave numbers ξ , no Fourier component of the solution is ever allowed to grow. The practical tool for verifying this is von Neumann analysis, a Fourier-based method that decomposes the numerical solution into wave modes of the form [4, 26]:

$$U_j^n = g^n e^{i\xi j \Delta x} \quad (2.16)$$

where $g(\xi)$ is the amplification factor for wave number ξ . Substituting this into the difference equation gives an explicit expression for $g(\xi)$. The scheme is stable if and only if:

$$|g(\xi)| \leq 1 \quad \text{for all } \xi \quad (2.17)$$

This is the condition we will verify for each time-stepping method in the sections that follow.

2.1.4 Algorithmic Framework: Time-Stepping Schemes

Explicit Schemes

Once the Schrödinger equation is spatially discretized into the matrix system (2.12), what remains is a system of coupled ordinary differential equations in time, one equation per grid point, all interacting through the Hamiltonian matrix H . The question is now how to advance this system forward in time. Several algorithmic strategies exist [19], and they differ in how they evaluate the right-hand side at each step.

Forward-Time Centered-Space (FTCS): The Naive Explicit Method The simplest approach is to apply the forward difference formula directly to the time

derivative. The new wave function is computed entirely from the current one [5]:

$$\psi(t + \Delta t) = \psi(t) - iH\Delta t \psi(t) = (I - iH\Delta t)\psi(t) \quad (2.18)$$

At each time step, the wave function is updated by a single matrix–vector multiplication. No system of equations needs to be solved, the future state is computed directly and explicitly from the current one. This is what makes the method explicit: we know ψ at time t , and we directly calculate ψ at time $t + \Delta t$.

For accuracy, Taylor expanding $\psi(t + \Delta t)$ shows the error per step is

$$\mathcal{O}(\Delta t^2),$$

which accumulates to

$$\mathcal{O}(\Delta t)$$

over $T/\Delta t$ total steps, first-order accuracy in time. The spatial discretization contributes

$$\mathcal{O}(\Delta r^2)$$

from the centered formula, giving second-order accuracy in space.

Even though its simplicity, FTCS is completely unusable for the Schrödinger equation, not because of poor accuracy, but because of unconditional instability. The Hamiltonian H is Hermitian with real eigenvalues λ , so the amplification factor for each Fourier mode is:

$$|g| = |1 - i\lambda\Delta t| = \sqrt{1 + \lambda^2\Delta t^2} > 1 \quad (2.19)$$

This is always strictly greater than one, for any eigenvalue and any time step. Every single Fourier component of the wave function grows at every step, no matter how small Δt is. The solution physically explodes. No choice of grid parameters

can fix this.

The Leapfrog Method When FTCS fails because it only looks forward in time, leapfrog fixes this by using a centered time difference which combines past and present [19, 4]:

$$\psi(t + \Delta t) = \psi(t - \Delta t) - 2iH\Delta t \psi(t) \quad (2.20)$$

This is a three-level method, it requires the wave function at two previous time levels, t and $t - \Delta t$, to compute the next one. It leaps over the current step using a centered derivative.

The centered structure gives second-order accuracy in time, the error is

$$\mathcal{O}(\Delta t^2).$$

Unlike FTCS, leapfrog can be stable, but only conditionally: the time step must satisfy

$$\Delta t \leq \frac{1}{\lambda_{\max}},$$

where $\lambda_{\max}$ is the largest eigenvalue of H .

However, even when stable, the method is only marginally so. The amplification factor satisfies

$$|g| = 1$$

exactly, errors neither grow nor decay. Nothing explodes, but nothing is corrected either. All modes propagate without damping, but not all at the correct speed, which causes wave packets to gradually drift out of phase. This marginal stability also becomes genuine instability when the potential varies or the problem becomes more complex.

Visscher's Staggered-Time Algorithm FTCS is unstable and leapfrog is fragile. This dilemma arises from treating the wave function as a single complex object.

Splitting it into real and imaginary parts reveals a better structure [5].

Writing

$$\psi = R + iI$$

where R and I are the real and imaginary parts, and substituting into the Schrödinger equation gives two coupled real equations:

$$\frac{\partial R}{\partial t} = HI \quad (2.21)$$

$$\frac{\partial I}{\partial t} = -HR \quad (2.22)$$

R evolves driven by I , and I evolves driven by R , but neither drives itself, a structure analogous to position and velocity in classical mechanics. This suggests defining R and I at staggered times instead of simultaneously [5]. R is defined at full time steps

$$0, \Delta t, 2\Delta t, \dots$$

and I at half steps

$$\frac{1}{2}\Delta t, \frac{3}{2}\Delta t, \dots$$

The discretized evolution equations are then [5]:

$$R(t + \Delta t) = R(t) + \Delta t H I(t + \frac{1}{2}\Delta t) \quad (2.23)$$

$$I(t + \frac{1}{2}\Delta t) = I(t - \frac{1}{2}\Delta t) - \Delta t H R(t) \quad (2.24)$$

Each update uses values symmetrically around the current time, a centered scheme, giving second-order accuracy in time while remaining fully explicit. No matrix inversion is needed. Each step is a direct computation: update R using I , then update I using R .

For probability conservation, since R and I are not defined at the same time, the probability density is defined as:

$$P(x, t) = R(x, t)^2 + I(x, t + \tfrac{1}{2}\Delta t) I(x, t - \tfrac{1}{2}\Delta t) \quad (2.25)$$

at integer times, with a symmetric counterpart at half-integer times. One can show directly that the discrete time derivative of the total probability vanishes identically, probability is exactly conserved at every step [5].

The method is conditionally stable, the time step must satisfy

$$\Delta t \leq \frac{m\Delta r^2}{1 + \frac{1}{2}m\Delta r^2 V_{\max}} \quad (2.26)$$

For any reasonable grid parameters this condition is easily satisfied, and the solution remains bounded and physical. Visscher's algorithm achieves what FTCS and leapfrog could not: a fully explicit method that is second-order accurate, exactly probability-conserving, and stably applicable to the Schrödinger equation [5].

Implicit Schemes

In explicit methods, the future state ψ^{n+1} is computed directly from the current state ψ^n , no system of equations needs to be solved. Implicit methods work differently, the future state appears on both sides of the equation, which means it cannot be computed directly. In the matrix setting, this means solving a linear system of the form

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

for ψ^{n+1} at every time step [19]. This increases the computational cost per step compared to explicit updates, but provides a major advantage in return and it is the ability to remain stable even for much larger time steps.

Backward Euler Instead of using the present state to move forward, the backward Euler method bases the update on the future itself [19, 25]. Rather than predicting where the wave function will be next, we ask if it ends up at $\psi(t + \Delta t)$, would that be consistent with the physics? We solve backwards from the answer:

$$\psi(t + \Delta t) = (I + iH\Delta t)^{-1}\psi(t) \quad (2.27)$$

This requires inverting the matrix $(I + iH\Delta t)$ at each step. The reward is unconditional stability, the amplification factor always satisfies

$$|g| \leq 1,$$

so nothing ever blows up, no matter how large the time step is.

However, because the forward and backward operators are not symmetric, the method is not unitary. At each step, the wave function shrinks slightly, which means total probability decreases over time. Even though the method is numerically safe, it does not respect the physics of quantum mechanics where probability must be conserved. For this reason, backward Euler is not used in practice for the Schrödinger equation. It serves as a stepping stone toward better methods like Crank–Nicolson.

The Crank–Nicolson Method Crank–Nicolson fixes backward Euler’s non-unitarity by symmetrizing [3]. Instead of evaluating $H\psi$ at either endpoint of the time interval, it averages over both, evaluating the right-hand side at the midpoint $t + \frac{1}{2}\Delta t$. This leads to a centered time discretization. Starting from the Schrödinger equation and approximating the time derivative with a centered difference [3]:

$$\frac{\psi^{n+1} - \psi^n}{\Delta t} = -\frac{i}{2}H(\psi^{n+1} + \psi^n) \quad (2.28)$$

Rearranging and collecting ψ^{n+1} on the left and ψ^n on the right gives:

$$\underbrace{\left(I + \frac{i\Delta t}{2}H\right)}_A \psi^{n+1} = \underbrace{\left(I - \frac{i\Delta t}{2}H\right)}_B \psi^n \quad (2.29)$$

The right-hand side $B\psi^n$ is fully known at each step. Solving for ψ^{n+1} requires inverting A . Since H is tridiagonal, each grid point couples only to its immediate neighbors. A is also tridiagonal, taking the form:

$$A = \begin{pmatrix} b_1 & c_1 & 0 & \cdots & 0 \\ a_2 & b_2 & c_2 & \cdots & 0 \\ 0 & a_3 & b_3 & \cdots & 0 \\ \vdots & & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & a_N & b_N \end{pmatrix} \quad (2.30)$$

where the diagonal entries are

$$b_i = 1 + \frac{i\Delta t}{2} \left(\frac{\hbar^2}{m\Delta r^2} + V_i \right),$$

and the off-diagonal entries are

$$a_i = c_i = -\frac{i\Delta t\hbar^2}{4m\Delta r^2},$$

with zeros everywhere else.

Tridiagonal systems of N equations can be solved in $\mathcal{O}(N)$ operations using the Thomas algorithm, a specialized form of Gaussian elimination that exploits this three-diagonal structure by sweeping forward to eliminate the lower diagonal and then backward to recover the solution [25]. This makes Crank–Nicolson as fast per step as an explicit method.

Algorithm Steps At each time step, the Crank–Nicolson algorithm proceeds as follows. First, the right-hand side $B\psi^n$ is computed directly, this is just a matrix-vector multiplication with the known current wave function, costing $\mathcal{O}(N)$ operations since B is tridiagonal. The result is a vector

$$\mathbf{b} = B\psi^n.$$

Second, the linear system

$$A\psi^{n+1} = \mathbf{b}$$

is solved for the unknown ψ^{n+1} using the Thomas algorithm, which works in two passes: a forward sweep that eliminates the lower diagonal entry in each row by subtracting a multiple of the row above, and a backward substitution that recovers the solution from the last grid point back to the first [25]. The entire solve costs $\mathcal{O}(N)$ operations.

Third, the recovered ψ^{n+1} becomes the new current state, and the process repeats for the next time step. The algorithm therefore marches forward in time one step at a time, each step requiring one matrix-vector product and one tridiagonal solve.

The key advantage of Crank–Nicolson over all explicit methods is that its time evolution operator is unitary, it preserves the norm of the wave function exactly at every time step, so total probability is always conserved [3]. This follows directly from the symmetric averaging structure. The amplification factor per eigenmode satisfies

$$|g| = \left| \frac{1 - i\lambda\Delta t/2}{1 + i\lambda\Delta t/2} \right| = 1 \quad (2.31)$$

exactly, for any real eigenvalue λ of H and any choice of Δt . The numerator and denominator are complex conjugates, so their ratio always has magnitude one. No mode ever grows or decays numerically.

This unitarity also implies unconditional stability, there is no restriction relating Δt to Δr . Unlike explicit methods whose stability regions exclude the imaginary

axis where the Schrödinger eigenvalues live, the Crank–Nicolson stability region covers the entire left half of the complex plane, a property known as A-stability [19]. This makes it well suited for stiff problems which are situations where the system contains widely different time scales, forcing explicit methods to take unreasonably small time steps just to remain stable, while Crank–Nicolson handles any step size without difficulty.

Intuitively, explicit schemes add energy to the system causing explosion, backward Euler removes energy causing fading, and Crank–Nicolson adds and removes nothing, it is perfectly balanced.

Runge–Kutta Methods Rather than computing the derivative once per time step, Runge–Kutta methods evaluate it at several carefully chosen intermediate points within the step and combine these estimates with specific weights to achieve higher-order accuracy [27]. The most widely used variant, RK4, uses four evaluations per step:

$$\psi^{n+1} = \psi^n + \frac{\Delta t}{6}(k_1 + 2k_2 + 2k_3 + k_4) \quad (2.32)$$

where k_1 is the slope at the start, k_2 and k_3 are improved midpoint estimates, and k_4 is the slope at the end. The midpoint estimates are weighted twice as heavily, reflecting their greater representativeness of the full interval. This yields a global error of $\mathcal{O}(\Delta t^4)$ which is a good improvement over second-order methods [27].

But despite its high accuracy, RK4 remains an explicit method and cannot be A-stable. Its stability region along the imaginary axis is finite, imposing a time step restriction. However, since it requires no linear system solve, all four stage computations are explicit matrix-vector products. It remains competitive when stiffness is moderate and the time step restriction is not prohibitive.

Comparison of Time-Stepping Algorithms

Method	Type	Accuracy	Stability	Unitary	Prob. Cons.	CFL
FTCS	Explicit	$\mathcal{O}(\Delta t)$	Never	No	No	None
Leapfrog	Explicit	$\mathcal{O}(\Delta t^2)$	Marginal	No	No	$\Delta t \leq 1/\lambda_{\max}$
Visscher	Explicit	$\mathcal{O}(\Delta t^2)$	Conditional	Yes	Yes	Eq. (2.26)
Backward Euler	Implicit	$\mathcal{O}(\Delta t)$	Unconditional	No	No	None
Crank-Nicolson	Implicit	$\mathcal{O}(\Delta t^2)$	Unconditional	Yes	Yes	None
RK4	Explicit	$\mathcal{O}(\Delta t^4)$	Conditional	No	No	$ \lambda \Delta t \lesssim 2.83$

Table 2.1: Comparison of time-stepping schemes for the time-dependent Schrödinger equation. Accuracy refers to the global error in time. Stability indicates whether errors remain bounded. Unitary indicates whether the evolution operator preserves the norm of ψ exactly. Prob. Cons. indicates whether total probability is exactly conserved at every step. CFL gives the time step restriction required for stability, where $\lambda_{\max}$ is the largest eigenvalue of H .

2.1.5 Limitations of the Finite Difference Method

Stability and the CFL Condition Explicit methods are not unconditionally stable. For the Schrödinger equation, the time step must satisfy a stability restriction often referred to as a CFL-type condition [28]:

$$\Delta t \leq \frac{m \Delta r^2}{\hbar + \frac{1}{2} m \Delta r^2 V_{\max}} \quad (2.33)$$

Refining the spatial grid therefore forces a quadratically smaller time step. Most of this extra computation is spent controlling numerical noise from rapidly oscillating short-wavelength components that carry no physical information which is a computationally wasteful consequence of the grid structure.

Implicit methods like Crank–Nicolson lift this restriction entirely, but at a differ-

ent cost. Despite unconditional stability and exact probability conservation, Crank–Nicolson requires solving a linear system at every time step. Even though the tridiagonal structure keeps this cost at $\mathcal{O}(N)$, the method becomes increasingly demanding in memory and CPU time as the system size grows. Furthermore, unconditional stability does not guarantee accuracy. If Δt is taken too large, the solution may remain bounded but accumulate significant phase errors that are physically incorrect.

The Curse of Dimensionality The deepest limitation of FDM is geometric. The method requires an explicit grid covering the entire computational domain. In 1D this is straightforward, but the wave function of an n -particle quantum system depends on $3n$ spatial variables. The number of grid points therefore grows exponentially with the number of particles, a system of just ten particles requires a grid in thirty dimensions, which no classical computer can store or process [29]. This makes FDM intractable for any realistic many-body quantum problem, regardless of the time-stepping scheme chosen.

The Inverse Problem For the inverse problem, recovering an unknown potential $V(x)$ from observations of ψ , FDM offers no direct solution [22]. One must instead resort to iterative schemes that solve the forward problem repeatedly, adjusting V at each step until the computed wave function matches the observed data. This is computationally expensive, sensitive to measurement noise, and scales poorly with system size. The structural limitations of the grid-based approach apply with full force in the inverse setting.

Taken together, these constraints, the CFL condition, the cost of implicit solvers, the exponential scaling with dimension, and the absence of a natural inverse problem framework all reveal the fundamental boundaries of classical grid-based methods for quantum problems. The following section addresses the inverse problem directly, introducing Tikhonov regularization as the standard classical strategy for recovering unknown potentials from noisy observations of the wave function.

2.2 Classical Inverse Solver: Tikhonov Regularization

2.2.1 Core Idea: Problem Formulation and the Forward Operator

In the forward problem, the potential $V(x)$ is known and the Schrödinger equation is used to compute the resulting wave function $\psi(x, t)$. The potential defines the landscape in which the quantum particle moves, and the Schrödinger equation determines how the particle evolves within it. The forward problem can therefore be viewed as the map

$$V(x) \mapsto \psi(x, t),$$

where the potential determines the resulting quantum evolution.

The inverse problem reverses this logic [7]. The potential is no longer given. Instead, only observations of the wave function at different times are available, possibly corrupted by measurement noise, and the goal is to determine what kind of potential could have produced this evolution. The inverse problem therefore attempts to invert the map

$$\psi(x, t) \mapsto V(x),$$

recovering the unknown potential from observations of the wave function [22]. In quantum experiments, the potential is rarely directly accessible. What is typically measured are energy levels, probability densities, or scattering data [30, 31]. For the one-dimensional harmonic oscillator studied in this thesis, the forward problem takes the quadratic potential

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

and computes how a quantum particle evolves within it. The inverse problem asks

the converse: given only measurements of the wave function over time, can one reconstruct that parabolic potential and recover the frequency ω ?

To formulate this mathematically, we introduce:

The Forward Operator. The forward operator A is the map that takes a potential $V(x)$ as input and returns the predicted wave function ψ as output. In this thesis, A is concretely the Crank–Nicolson solver developed in Section 2.2: given a candidate potential, it performs the full discrete time evolution and returns the predicted wave function values at the observation points.

The inverse problem is therefore posed as finding a potential V such that

$$A(V) \approx \psi^{\text{obs}} \quad (2.34)$$

This operator is highly nonlinear. A small perturbation in the potential can significantly alter the wave function through interference effects, phase shifts, and oscillatory interactions. Unlike a simple algebraic equation, the forward operator cannot generally be inverted analytically. Furthermore, the mapping from V to ψ is not one-to-one: different potentials may produce wave functions that are nearly indistinguishable within the precision of the measurements. The inverse problem therefore attempts to recover information that is partially lost in the forward evolution.

In practice, the observed wave function is never exact. Measurements always contain noise, which we model as additive Gaussian noise:

$$\psi^{\text{obs}} = \psi^{\text{true}} + \sigma\eta, \quad \eta \sim \mathcal{N}(0, I) \quad (2.35)$$

where σ controls the noise level. Here η is drawn from the standard normal distri-

bution, whose probability density function is

$$f(\eta) = \frac{1}{\sqrt{2\pi}} e^{-\eta^2/2} \quad (2.36)$$

This density, shown in Figure 2.1, is symmetric around zero and takes both positive and negative values with equal probability. Consequently, the perturbation $\sigma\eta$ does not only add to the true signal: at points where $\eta < 0$, the noise is effectively subtracted from ψ^{true} .

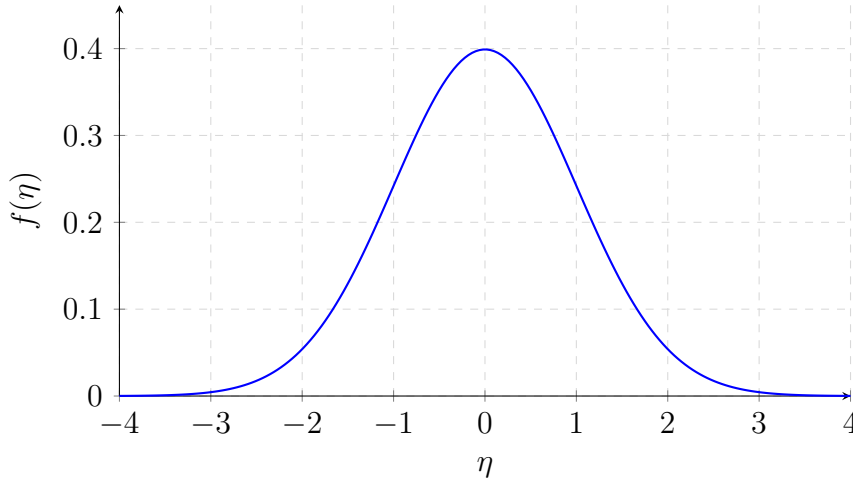


Figure 2.1: Probability density function of the standard normal distribution $\mathcal{N}(0, 1)$ used to model the measurement noise. The density is symmetric around zero, so noise samples are equally likely to be positive or negative.

The additive model is not the only possibility. In some measurement settings the noise instead acts multiplicatively,

$$\psi^{\text{obs}} = \psi^{\text{true}} (1 + \sigma\eta), \quad (2.37)$$

so that the perturbation scales with the signal amplitude itself. Multiplicative noise is typically handled by first filtering or transforming the data (for instance, a logarithmic transformation converts it into an approximately additive perturbation, after which standard denoising filters apply) before the inversion is performed. In this work, only the additive model of Equation (2.35) is considered.

In the synthetic setting considered in this thesis, ψ^{true} is generated numerically by solving the Schrödinger equation with the true harmonic potential

$$V^{\text{true}}(x) = \frac{1}{2}m\omega^2 x^2,$$

after which artificial Gaussian noise is added to simulate realistic measurement conditions.

The inverse problem then becomes: pretending the potential were completely unknown, can it be reconstructed reliably from the noisy observations ψ^{obs} alone?

Everything that follows, like ill-posedness, regularization, and optimization, is devoted to answering this question reliably.

2.2.2 Mathematical Foundations

Hadamard's Three Conditions

A mathematical problem can be viewed as a mapping: data enters, and a solution emerges. According to Hadamard, a problem is said to be well-posed if it satisfies three conditions simultaneously [18, 6]:

1. **Existence:** a solution exists.
2. **Uniqueness:** the solution is unique.
3. **Continuous dependence:** small perturbations in the input produce only small perturbations in the output.

The third condition is also called stability. It is the most physically important of the three because experimental data is never exact, measurements always contain noise. A problem that violates continuous dependence is called ill-posed [18].

Ill-Posedness and Instability

For the inverse Schrödinger problem, existence is generally not the main difficulty, and uniqueness can often be established under suitable assumptions and sufficient data. The condition that fundamentally fails is continuous dependence [7].

To see why, suppose two potentials $V_1(x)$ and $V_2(x)$ differ by a rapidly oscillating perturbation:

$$V_2(x) = V_1(x) + \epsilon \sin(nx)$$

For small ϵ and large n , the perturbation has a very small amplitude but extremely high frequency. Passing both potentials through the forward operator gives

$$\psi_1 = A(V_1), \quad \psi_2 = A(V_2).$$

Although the potentials differ significantly in their oscillatory structure, the corresponding wave functions may remain extremely close. The forward evolution tends to be far less sensitive to highly oscillatory perturbations in the potential than to large-scale features. As a result, distinct potentials can produce nearly indistinguishable observations.

Now reverse the problem. Suppose only the observed wave function ψ^{obs} is available. The reconstruction algorithm must determine which potential produced it. But if many different potentials generate nearly identical wave functions, then even a tiny perturbation in the observed data may drastically change the reconstructed solution.

For stability, continuous dependence would require an estimate of the form

$$\|V_1 - V_2\| \leq C \|A(V_1) - A(V_2)\|$$

for some constant C . This inequality would guarantee that nearby wave functions correspond to nearby potentials.

For the inverse Schrödinger problem, however, no such uniform constant generally exists. The ratio

$$\frac{\|V_1 - V_2\|}{\|A(V_1) - A(V_2)\|}$$

can become arbitrarily large for highly oscillatory perturbations. This is the mathematical signature of ill-posedness [7, 20].

This instability is not caused by numerical discretization or poor algorithmic choices. It is a property of the inverse problem itself, the forward evolution does not preserve all information about the potential in a stably recoverable form.

Noise Amplification

The practical consequence of ill-posedness is noise amplification [20]. High-frequency perturbations in the potential often produce only extremely small changes in the observed wave function. Recovering these components therefore requires amplifying tiny differences in the data.

Heuristically, the forward operator attenuates highly oscillatory modes. In Fourier language, one may think schematically of

$$A(e^{inx}) \approx e^{-n} e^{inx},$$

meaning that high-frequency modes become strongly suppressed in the forward direction.

A naive inversion would therefore behave roughly as

$$A^{-1}(e^{inx}) \approx e^n e^{inx},$$

which exponentially amplifies high-frequency components.

If the observed data contains noise of amplitude σ at frequency n , the inversion amplifies that noise by approximately e^n . Even extremely small measurement

errors can therefore dominate the reconstruction at high frequencies. Instead of recovering a smooth physical potential, the solution becomes violently oscillatory and dominated by noise.

This is analogous to increasing the volume of a noisy audio recording, both the signal and the noise are amplified, but eventually the noise overwhelms the information content entirely.

Discretization of the Forward Operator

To formulate the inverse problem computationally, the continuous forward operator A must be expressed in discrete form that a computer can handle. This follows directly from the spatial and temporal discretization already developed in Section 2.1. The spatial domain is replaced by a grid of N points, and the wave function at each time step becomes a vector of N complex values, as in equation (2.13). The potential $V(x)$ similarly becomes a discrete vector:

$$\mathbf{v} = (V(x_1), V(x_2), \dots, V(x_N))^T$$

The Schrödinger equation becomes the matrix system of equation (2.13), with the Hamiltonian $H = T + U$ structured exactly as in equation (2.11): the kinetic energy T contributes the tridiagonal coupling between neighboring points, and the potential energy U sits entirely on the diagonal, one value per grid point.

The unknown potential enters the Hamiltonian only through the diagonal entries. Changing v means changing the diagonal of H , which changes both sides of the Crank-Nicolson linear system (2.29), which in turn changes the solution ψ^{n+1} in a nonlinear way. This is why A depends on v nonlinearly, it is not a simple matrix multiplication, but a full time evolution whose internal structure is coupled to the unknown.

At each time step, the Crank-Nicolson scheme advances the wave function by

solving the system of equation (2.29).

Applying this repeatedly from $t = 0$ to $t = T$ over M time steps constitutes the discrete forward operator A . It takes the initial wave function ψ^0 and the potential vector v , runs the full time evolution, and returns the predicted wave function at the observation points:

$$A(\mathbf{v}) = \text{CN}_M \circ \dots \circ \text{CN}_1(\psi^0, \mathbf{v})$$

where each CN_k denotes one Crank-Nicolson step with the Hamiltonian built from $\mathbf{v}$.

The Noisy Observation Model In any real experiment, the wave function is never observed exactly. Every measurement carries noise. The observation model introduced in equation (2.35) gives:

$$\psi^{\text{obs}} = A(\mathbf{v}^{\text{true}}) + \sigma \cdot \eta, \quad \eta \sim \mathcal{N}(0, I)$$

where η is a vector of independent standard normal random numbers representing pure white noise, and σ scales the noise amplitude. The inverse problem is then: given only ψ^{obs} and σ , find v .

Failure of Direct Inversion and Moore-Penrose Pseudoinverse The natural instinct is to write $\mathbf{v} \approx A^{-1}(\psi^{\text{obs}})$ and simply invert A . This fails for two fundamental reasons.

Nonlinearity. A is not a linear map from $\mathbf{v}$ to ψ . Changing $\mathbf{v}$ changes the Hamiltonian matrix, which appears inside the Crank-Nicolson system in a nonlinear way. There is no matrix M such that $A(\mathbf{v}) = M\mathbf{v}$ for all $\mathbf{v}$, so standard matrix inversion does not apply.

The Moore-Penrose pseudoinverse is discontinuous [32]. Even if A is linearized

around some point, the resulting linear map is compact, it maps the infinite-dimensional space of potentials into a much smaller space of wave functions, collapsing information [17].

Definition (Moore-Penrose pseudoinverse). The Moore-Penrose pseudoinverse $A^\dagger$ is the generalized inverse of an operator A , defined as the least-squares solution operator for $A\mathbf{v} = \psi$ when an exact inverse does not exist. It provides the solution of minimal norm among all possible least-squares solutions [32].

For compact operators, $A^\dagger$ is unbounded [17, 7], meaning:

$$\|A^\dagger\psi_1 - A^\dagger\psi_2\| \text{ can be arbitrarily large even when } \|\psi_1 - \psi_2\| \text{ is tiny}$$

A tiny noise perturbation $\sigma\eta$ in ψ^{obs} is therefore amplified enormously when $A^\dagger$ is applied. This is the ill-posedness established in the previous section, now expressed in the discrete matrix language of the Crank-Nicolson solver.

Since direct inversion is impossible, a different strategy is needed. Instead of asking which $\mathbf{v}$ exactly satisfies $A(\mathbf{v}) = \psi^{\text{obs}}$, which is a question with no stable answer, we ask which $\mathbf{v}$ approximately satisfies it while also being physically reasonable. This is the variational approach formalized by Tikhonov regularization, developed in the following section [8, 33].

2.2.3 Tikhonov Regularization

The Regularized Functional

Rather than inverting A directly, Tikhonov regularization replaces the ill-posed inversion with a tractable minimization problem [8]. Instead of asking which v exactly satisfies

$$A(v) = \psi^{\text{obs}},$$

we ask which v best balances two competing demands simultaneously [8, 9]:

$$\min_V \left\{ \underbrace{\|A(V) - \psi^{\text{obs}}\|^2}_{\text{data fidelity}} + \lambda \underbrace{\|V\|^2}_{\text{regularization}} \right\} \quad (2.38)$$

This is the quantity the algorithm minimizes. The two terms serve complementary roles.

The data fidelity term $\|A(V) - \psi^{\text{obs}}\|^2$ measures how well the candidate potential explains the observations. Given a candidate V , the forward solver produces a predicted wave function $A(V)$, which is then compared to ψ^{obs} . If they are very different, the error is large. If they match, the error is small. Minimizing this term alone is not sufficient, the optimizer becomes obsessed with fitting every tiny fluctuation in the noisy data, including random noise, producing wildly oscillating, non-physical reconstructions. This is overfitting the noise.

The regularization term $\lambda\|V\|^2$ penalizes potentials that are unnecessarily large or rapidly oscillating. It tells the algorithm: among all potentials that explain the data reasonably well, prefer the smoothest and most physically plausible one. Minimizing this term alone is useless, it always returns $V = 0$ regardless of the data. But combined with the first term, it acts as a physical constraint. The optimizer cannot drive the fidelity term to zero by fitting the noise, because doing so would require a wildly oscillatory potential that drives the regularization term to infinity. The two terms compete, and their balance yields a stable, physically meaningful solution.

The Role of λ

The regularization parameter $\lambda > 0$ controls the balance between data fidelity and solution smoothness:

Regime	Behaviour	Result
$\lambda \rightarrow 0$	Penalty vanishes, fidelity dominates	Unstable inversion, noise amplified
$\lambda \rightarrow \infty$	Penalty dominates, data ignored	$V \rightarrow 0$, structure lost
Optimal λ	Balanced compromise	Smooth, physically plausible reconstruction

For the harmonic oscillator, the optimal λ produces a reconstruction that is recognizably quadratic, not a perfect parabola, but one that captures the shape of

$$\frac{1}{2}m\omega^2 x^2.$$

Well-Posedness of the Regularized Problem

Recalling Hadamard's third condition: the solution must depend continuously on the data. The Tikhonov functional restores this property [8, 9]. For any fixed $\lambda > 0$, the minimizer V_λ depends continuously on ψ^{obs} , a small perturbation in the observed data produces only a small change in the reconstruction. The penalty term stabilizes the otherwise ill-conditioned inversion by preventing noise amplification.

The minimizer V_λ satisfies the Euler-Lagrange condition:

$$J_A^T (A(V_\lambda) - \psi^{\text{obs}}) + \lambda V_\lambda = 0 \quad (2.39)$$

where J_A^T is the transpose of the Jacobian of A , which maps the mismatch between prediction and observation back into the space of potentials. The first term drives correction in the direction of reduced error, while the second term suppresses large or oscillatory solutions. For any $\lambda > 0$, this yields a unique and stable solution, resolving ill-posedness [9].

Choosing lambda: The Morozov Discrepancy Principle

The remaining question is how to choose λ without knowing the true potential. The Morozov discrepancy principle provides a principled answer [23]. Let $\delta = \sigma$ denote the known noise level. The principle states:

$$\|A(V_\lambda) - \psi^{\text{obs}}\| \approx \delta \quad (2.40)$$

If $\|A(V_\lambda) - \psi^{\text{obs}}\| \gg \delta$, the reconstruction is over-smoothed and misses structure in the data. If $\|A(V_\lambda) - \psi^{\text{obs}}\| \ll \delta$, the model is fitting noise. The optimal choice satisfies the balance where the residual matches the noise level [23].

In practice, one evaluates λ on a logarithmic grid (e.g. 10^{-6} to 10^2), computes $\|A(V_\lambda) - \psi^{\text{obs}}\|$ for each value, and selects the λ where the residual crosses σ . For the harmonic oscillator, this choice of λ typically coincides with the one minimizing the reconstruction error $\|V_\lambda - V^{\text{true}}\|$.

2.2.4 Solution Algorithm

Overview

The core idea of the inverse algorithm is simple: start with a wrong guess for the potential, simulate the wave function using the forward solver, compare it to the observed wave function, and gradually modify the potential until the simulation matches the observation. The inverse method is, at its heart, a loop: guess, simulate, compare, correct, repeat.

Building the Discrete System

The algorithm operates entirely on the discrete structures developed in Section 2.1. The spatial domain is divided into N grid points. The wave function becomes a vector of N complex values, the potential becomes a vector $\mathbf{v} = (V_1, \dots, V_N)^T$,

and the Schrödinger equation becomes the matrix system of equations (2.12) and (2.13).

The Hamiltonian is assembled from the kinetic energy matrix T of equation (2.8) and the potential energy matrix U of equation (2.9):

$$H(\mathbf{v}) = T + U(\mathbf{v})$$

The kinetic energy matrix T is fixed, it depends only on the grid spacing and carries the tridiagonal coupling between neighboring points. The potential energy matrix U is diagonal, with V_i on the i -th diagonal entry. This means the unknown potential enters the Hamiltonian only through the diagonal. During the inverse problem, modifying the potential means modifying only the diagonal entries of H — nothing else changes.

The Crank-Nicolson scheme then produces two matrices from $H(\mathbf{v})$, as in equation (2.29):

$$A_{\text{CN}} = I + \frac{i\Delta t}{2}H(\mathbf{v}), \quad B_{\text{CN}} = I - \frac{i\Delta t}{2}H(\mathbf{v})$$

At each time step, the forward solver advances the wave function by first computing the right-hand side $B_{\text{CN}}\psi^n$, then solving for the next state:

$$\psi^{n+1} = A_{\text{CN}}^{-1}(B_{\text{CN}}\psi^n)$$

Repeating this over all time steps constitutes the discrete forward operator $A(\mathbf{v})$ of equation (2.40).

Generating Synthetic Data

Before the inverse problem begins, synthetic observed data must be generated. The true potential $V^{\text{true}}(x) = \frac{1}{2}m\omega^2x^2$ is used with the Crank-Nicolson solver to produce the noiseless wave function ψ^{true} . Gaussian noise is then added following

the observation model of equation (2.35):

$$\psi^{\text{obs}} = \psi^{\text{true}} + \sigma \cdot \eta, \quad \eta \sim \mathcal{N}(0, I)$$

The parameter σ controls the noise strength. A value of $\sigma = 0.001$ produces nearly clean data, while $\sigma = 0.2$ corresponds to heavily corrupted measurements. From this point, the true potential is set aside. The inverse algorithm has access only to ψ^{obs} and σ .

Iterative Optimization

The minimization of the Tikhonov functional (2.42) is carried out iteratively using gradient descent. Starting from an initial guess $V^{(0)}$, which can be zero everywhere, a constant, or a smooth random curve, the potential is updated at each iteration by descending along the gradient of the loss:

$$V^{(k+1)} = V^{(k)} - \alpha \nabla_V \mathcal{L}(V^{(k)}) \quad (2.41)$$

where α is the learning rate and $\mathcal{L}(V)$ is the Tikhonov functional of equation (2.42). The gradient has two components, one for each term of the loss.

Gradient of the Regularization Term The gradient of the penalty term is straightforward. Since $\lambda \|V\|^2 = \lambda(V_1^2 + V_2^2 + \dots + V_N^2)$, differentiating with respect to each V_i gives:

$$\nabla_V (\lambda \|V\|^2) = 2\lambda V \quad (2.42)$$

The penalty gradient is linear in V and always points toward zero — it pushes the potential toward a smoother, smaller solution at every step.

Gradient of the Data Fidelity Term The gradient of the residual term requires the Jacobian of A with respect to V :

$$\nabla_V \|A(V) - \psi^{\text{obs}}\|^2 = 2J_A^T(A(V) - \psi^{\text{obs}}) \quad (2.43)$$

The Jacobian J_A is the matrix that encodes the sensitivity of the forward solver's output to changes in the potential. Its entry (i, j) answers the question: if I nudge the potential at grid point j by a tiny amount, how much does the i -th component of the predicted wave function change? Written out explicitly, J_A is an $M \times N$ matrix where M is the number of observation points and N is the number of grid points:

$$J_A = \begin{pmatrix} \frac{\partial[A(V)]_1}{\partial V_1} & \frac{\partial[A(V)]_1}{\partial V_2} & \cdots & \frac{\partial[A(V)]_1}{\partial V_N} \\ \frac{\partial[A(V)]_2}{\partial V_1} & \frac{\partial[A(V)]_2}{\partial V_2} & \cdots & \frac{\partial[A(V)]_2}{\partial V_N} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial[A(V)]_M}{\partial V_1} & \frac{\partial[A(V)]_M}{\partial V_2} & \cdots & \frac{\partial[A(V)]_M}{\partial V_N} \end{pmatrix} \quad (2.44)$$

Each column j asks: if V_j is perturbed, how does the entire predicted wave function respond? Each row i asks: how does the i -th observation point react to changes across all potential entries? If changing V_j causes a large change in the wave function, the corresponding Jacobian entry is large. If the wave function barely reacts, the entry is small. This matrix therefore captures how the entire time evolution responds to local perturbations in the potential.

Computing J_A analytically would require differentiating through the entire Crank-Nicolson time loop with respect to every entry of V , which is algebraically heavy and implementation-intensive. Instead, numerical finite differences are used [25]. To compute the j -th column of J_A , the potential entry V_j is perturbed by a

small amount ϵ , the forward solver is rerun, and the change in output is observed:

$$\frac{\partial A}{\partial V_j} \approx \frac{A(V + \epsilon e_j) - A(V)}{\epsilon} \quad (2.45)$$

where e_j is the unit vector in the j -th direction. Concretely, if V_2 is nudged from V_2 to $V_2 + \epsilon$, the Crank-Nicolson solver is rerun with this perturbed potential. If the predicted wave function changes significantly, the Jacobian entry $\partial A / \partial V_2$ is large. If it barely changes, the entry is small. This procedure is repeated for each of the N entries of $\mathbf{v}$, costing N additional forward solves per gradient evaluation, expensive for large N , but perfectly manageable for the 1D harmonic oscillator.

Full Gradient and Update Combining both terms, the full gradient at iteration k is:

$$\nabla_V \mathcal{L}(V^{(k)}) = 2J_A^T (A(V^{(k)}) - \psi^{\text{obs}}) + 2\lambda V^{(k)} \quad (2.46)$$

In practice, this entire procedure is handled automatically by passing the loss function to `scipy.optimize.minimize` [34], which perturbs variables, estimates gradients numerically via finite differences, and updates the potential iteratively. The role of the implementation is simply to define the physics through the loss function.

Iteration Steps

At each iteration k , the algorithm proceeds as follows. First, the Crank-Nicolson solver is run with $V^{(k)}$ to obtain $A(V^{(k)})$. Second, the residual is computed:

$$r = A(V^{(k)}) - \psi^{\text{obs}}$$

Third, the gradient is assembled using equation (2.50). Fourth, the potential is updated according to equation (2.45). Fifth, the stopping criterion is checked: if $\|r\| \approx \sigma$, the Morozov condition (2.44) is satisfied and the algorithm stops. Other-

wise, the process repeats from step one. Each iteration costs one forward solve plus one gradient computation, and the potential is refined gradually toward a smoother, more physically plausible shape.

Step	Operation	Mathematical Object
1	Choose true potential	$V^{\text{true}}(x) = \frac{1}{2}m\omega^2x^2$
2	Run forward solver	$\psi^{\text{true}} = A(V^{\text{true}})$
3	Add noise	$\psi^{\text{obs}} = \psi^{\text{true}} + \sigma\eta$
4	Initialize guess	$V^{(0)} = 0$
5	Build Hamiltonian	$H(\mathbf{v}) = T + \text{diag}(\mathbf{v})$
6	Run Crank-Nicolson	$A(V^{(k)}) = \text{CN}_M \circ \dots \circ \text{CN}_1(\psi^0, V^{(k)})$
7	Compute residual	$r = A(V^{(k)}) - \psi^{\text{obs}}$
8	Compute loss	$\mathcal{L}(V) = \ r\ ^2 + \lambda\ V\ ^2$
9	Compute gradient	$g = 2J_A^T r + 2\lambda V^{(k)}$
10	Update potential	$V^{(k+1)} = V^{(k)} - \alpha g$
11	Check stopping criterion	$\ r\ \approx \sigma \quad (\text{Morozov})$
12	Output reconstruction	$V^{\text{rec}} \approx V^{\text{true}}$

Table 2.2: Summary of the Tikhonov inverse algorithm for the 1D harmonic oscillator. Steps 6 to 11 form the iterative loop, repeated until the Morozov discrepancy criterion is satisfied. The potential is recovered by adjusting the diagonal entries of the Hamiltonian until the predicted wave function matches the noisy observations.

The Three Regularization Regimes

As λ is varied and the optimization is run to convergence, three characteristic behaviors emerge in the reconstructed potential $V^{\text{rec}}(x)$:

Under-regularized ($\lambda \approx 0$): the penalty is negligible and the optimizer fits both the signal and the noise. The reconstructed potential develops rapid non-physical oscillations that perfectly explain ψ^{obs} but are physically nonsensical. The reconstruction error $\|V^{\text{rec}} - V^{\text{true}}\|$ is large.

Optimally regularized: the optimizer balances both terms. The residual settles near σ , satisfying the Morozov criterion (2.44). The reconstructed potential $V^{\text{rec}}(x)$ is a smooth curve that closely approximates the true parabola of the harmonic oscillator. The reconstruction error is minimized.

Over-regularized (large λ): the penalty term dominates. The optimizer drives V toward zero to minimize $\lambda\|V\|^2$, ignoring the data entirely. The reconstructed potential is nearly flat, and the quadratic shape of the harmonic oscillator is lost.

This three-regime analysis demonstrates the essential role of regularization in stabilizing the inverse problem and establishes the quantitative classical baseline against which the inverse PINN approach of Chapter 3 is compared.

2.2.5 Limitations of the Classical Inverse Approach

Noise Sensitivity Even with Tikhonov regularization, reconstruction quality degrades rapidly as the noise level σ increases. For small σ , the regularized solution closely approximates the true potential. As σ grows, however, the noise floor rises and the Morozov criterion (2.44) forces the algorithm to accept a larger residual, meaning less information about the true potential is recoverable. For severely corrupted data, the reconstruction may provide only a rough qualitative approximation of $V^{\text{true}}(x)$, regardless of the choice of λ .

Parameter Selection The regularization parameter λ is problem-dependent. The Morozov discrepancy principle provides a principled selection criterion, but it requires precise knowledge of the noise level $\delta = \sigma$, which is rarely available exactly in practice. When σ is only approximately known, the selected λ is itself approximate, and the reconstruction quality degrades accordingly. Alternative selection strategies such as the L-curve method [21] or generalized cross-validation [35], each introduce their own assumptions and computational overhead, and none provides a universally optimal rule.

Computational Cost Each evaluation of the forward operator $A(V)$ requires a complete Crank-Nicolson solve over all N grid points and M time steps. Computing the Jacobian J_A via finite differences, as in equation (2.49), requires an additional N forward solves per gradient evaluation. Convergence of the optimizer typically requires hundreds to thousands of iterations. The total cost therefore scales as $\mathcal{O}(N \cdot M \cdot K)$, where K is the number of iterations, making the approach increasingly expensive for fine spatial grids or long simulation windows.

Scalability The deepest structural limitation is dimensional. The Crank-Nicolson forward solver underlying $A(V)$ inherits all the dimensional constraints of FDM. For multi-particle quantum systems where the wave function lives in a $3n$ -dimensional configuration space, both the forward solve and the inverse optimization become computationally intractable [29]. The curse of dimensionality applies with full force in the inverse setting, not only must the grid grow exponentially, but the Jacobian J_A grows with it, making gradient computation infeasible.

Absence of a Unified Framework The classical approach treats the forward and inverse problems as two separate computational tasks. The forward solver is built first, and the inverse algorithm is wrapped around it as an outer optimization loop. This separation means that any improvement to one does not automatically

benefit the other, and the entire pipeline must be redesigned when the problem changes. There is no single formulation that merges both the physics and the inversion objective simultaneously.

2.2.6 Summary and Transition

Several structural insights emerge from the classical treatment developed in this chapter. The Hamiltonian matrix is tridiagonal because discretizing the second derivative in the Schrödinger equation couples each grid point only to its immediate neighbors, locality in physics becomes sparsity in linear algebra. Crank-Nicolson is unitary because its symmetric implicit-explicit structure preserves the norm of the wave function exactly for any Hermitian Hamiltonian, regardless of the time step. The inverse problem amplifies noise because reconstructing hidden causes from observed effects requires, in effect, dividing by small singular values of the forward operator, which magnifies measurement errors catastrophically. Tikhonov regularization resolves this by balancing data fidelity against solution smoothness through the parameter λ . And the optimizer works by adjusting the diagonal entries of the Hamiltonian, precisely the entries that encode the local potential energy values until the predicted quantum evolution matches the observed data.

The comparison between the classical inverse method and the Physics-Informed Neural Network approach developed in Chapter 3 is not perfectly fair because Tikhonov regularization requires many forward solves, one per iteration of the optimizer, while a PINN is trained once and encodes both the forward physics and the inverse objective simultaneously within a single loss function [11]. This asymmetry is not a flaw in the comparison, it is the thesis argument. The computational and structural cost of the classical approach, documented throughout this chapter, is precisely what motivates a unified framework that eliminates the need for repeated solver calls, explicit grid construction, and problem-specific regularization tuning. Chapter 3 develops that framework.

Chapter 3

Physics-Informed Neural Networks for Forward and Inverse Quantum Problems

The previous chapter established the classical numerical framework for both the forward and inverse Schrödinger problems. Finite difference methods provide rigorous tools for simulating quantum evolution, and Tikhonov regularization offers a principled strategy for recovering unknown potentials from noisy observations. Yet both approaches share limitations as discussed earlier.

This chapter introduces Physics-Informed Neural Networks as the alternative framework that addresses these limitations [11, 10, 12]. Before developing the PINN formulation, we briefly trace the conceptual path that leads here.

3.1 From Classical Solvers to Data-Driven Models

3.1.1 The Shift in Perspective

For most of the history of computational science, solving a problem meant specifying the exact procedure that produces its solution: an algorithm encodes a rule,

and the computer executes that rule precisely. This paradigm has been extremely successful when the governing equations are known explicitly, as in the finite difference methods discussed in Chapter 2. However, certain problems resist this approach, not because they are inherently more difficult, but because no explicit rule mapping inputs to outputs can be readily formulated. Examples include pattern recognition, image analysis, and highly complex nonlinear systems, where the underlying relationship exists but is not fully known.

Machine learning introduces a different perspective. Instead of explicitly programming the steps required to solve a problem, a model is provided with examples and learns the relationship between inputs and outputs directly from data [36]. During training, the model adjusts its internal parameters to reduce the difference between its predictions and the desired results. Once this learning process is complete, the model can make predictions for new inputs that were not part of the training data.

This data-driven approach offers an alternative to traditional algorithmic design. Instead of deriving a dedicated numerical procedure for every problem, the objective is to learn a general representation of the underlying mapping from the available information. As a result, machine learning has become a powerful tool for approximating complex functions and solving problems for which explicit solution strategies are difficult to construct.

3.1.2 Machine Learning: Core Concepts

Data-driven models cover a broad range of methods designed to perceive data, learn from it, and take actions toward a defined goal. Machine learning is the subfield concerned specifically with the learning component. Rather than following an explicitly programmed sequence of steps, a machine learning model adjusts a set of internal parameters, commonly called weights, based on data, in order to reduce the difference between its predictions and the desired outcome.

Machine learning encompasses a range of algorithm families including decision trees, nearest-neighbor methods, and clustering techniques, and each suited to different types of problems and data structures [36]. A common distinction is made between supervised learning, where the model learns from labeled examples consisting of known input-output pairs, and unsupervised learning, where the model identifies structure in data without explicit labels. This distinction becomes relevant later in this chapter, where Physics-Informed Neural Networks reduce the dependence on labeled data characteristic of supervised learning.

Two further concepts are central to all machine learning methods. First, the distinction between parameters, which are adjusted automatically during training (such as the weights of a model), and hyperparameters, which are chosen beforehand by the practitioner (such as the learning rate or the number of layers in a network). Second, the need for a quantitative measure of error: in regression problems, where the output is a continuous value rather than a discrete category, the mean squared error (MSE) is the most widely used metric, defined as the average of the squared differences between predicted and true values [36]. This metric reappears directly in the loss function of the neural networks discussed in the following section.

3.1.3 Evaluating Models

A central question in any machine learning approach is how to determine whether a trained model is good. The most direct measure is the training error itself, typically the mean squared error introduced previously. However, a model that fits its training data extremely well is not necessarily a good model, it may have simply memorized the data rather than learned the underlying structure, which is referred to as overfitting. Conversely, a model that is too simple to capture the relevant structure at all is said to underfit [36].

This trade-off is often described in terms of bias and variance [37]. A high-bias model is overly restrictive and may fail to capture important patterns present

in the data, leading to underfitting. A high-variance model, on the other hand, is excessively sensitive to the particular examples used during training and may perform poorly when presented with new data, leading to overfitting. Successful machine learning models therefore seek a balance between these two sources of error in order to achieve good generalization [37].

The concepts of model complexity, generalization, and error minimization are fundamental throughout machine learning and remain equally important for neural networks. With this framework in place, we now turn to the basic computational unit from which neural networks are constructed.

3.2 Neural Networks

3.2.1 Constructing a Neural Network in Practice

Before developing the mathematical formulation of a neural network, it is useful to describe what one looks like as an object that is built and run on a computer instead of treating it as a black box.

In Python, a neural network is a function. It takes a list of numbers as input and returns a list of numbers as output. Internally, it is built by composing several smaller functions called layers, each of which is also a function: it receives a list of numbers, transforms them, and returns a new list [36]. Each layer holds two objects: a matrix of weights and a list of biases, and uses them to compute a weighted combination of its inputs, followed by a fixed nonlinear transformation. Stacking several layers means calling them one after another, so that the output of the first becomes the input of the second, the output of the second becomes the input of the third, and so on. It is helpful to picture each layer as a separate room: the input enters room one, which processes it and passes a new list of numbers to room two, which does the same, until the final room returns the network's output. The network itself is therefore nothing more than a composition of these room functions,

and its only learnable content is the collection of weight matrices and bias lists held inside each room.

Deciding how many rooms there are, how many neurons each room contains, and what nonlinear transformation each room applies constitutes the network's architecture. These choices are fixed before training begins and determine the shape of every weight matrix and every bias list in the network.

Training requires two additional components. The first is a loss function: given the network's output and the desired result, it returns a single number measuring how far apart they are. The second is an optimizer: it uses information about how the loss depends on each weight and bias across all rooms in order to adjust them and reduce the error. The sequence that repeats until the network has learned, pass the input through all rooms to obtain an output, compute the loss, determine how each weight and bias contributed to the error, update them is called the training loop.

Writing all of this from scratch in Python would require implementing every layer function with its weight matrix and bias list, every nonlinear transformation, the composition of rooms into a network, and crucially, the procedure for determining how the loss depends on every single weight and bias across every room, a computation that grows rapidly in complexity as the number of rooms increases. A library such as PyTorch provides ready-made implementations of all of these components [38]. The architecture, the loss function, and the optimizer are still specified by the user. PyTorch only removes the need to implement every line by hand.

3.2.2 From Mesh to Function: The Role of a Single Neuron

It is useful to recall how the classical solvers of Chapter 2 represent a solution, in order to highlight the shift introduced by neural networks. In the Crank-Nicolson scheme, space and time were discretized into a mesh, and the equation was solved point by point on that mesh, the mesh itself was the solution. A neural network

takes a different approach. The wave function is treated as an unknown function, which is approximated by another function, the network itself, whose parameters are adjusted so that it matches the true wave function as closely as possible [10, 11]. The network takes the coordinates x and t as input and returns the wave function directly, with no mesh involved at any stage.

The inputs of the network are not vectors of data points, but variables at a single query point. In the present work, one position x and one time t . The network is evaluated at one point, returns a prediction at that point, and can be evaluated again at any other point in the domain, including points that were never part of any training set and do not lie on any grid. It's the reason why neural networks are described as mesh-free, the network is not a table of values defined only at fixed locations, but a function defined everywhere, which can be queried at arbitrary (x, t) [11].

The smallest building block through which this is realized is a single neuron [39, 40]. Figure 3.1 shows its structure schematically, for the case of the two inputs relevant here.

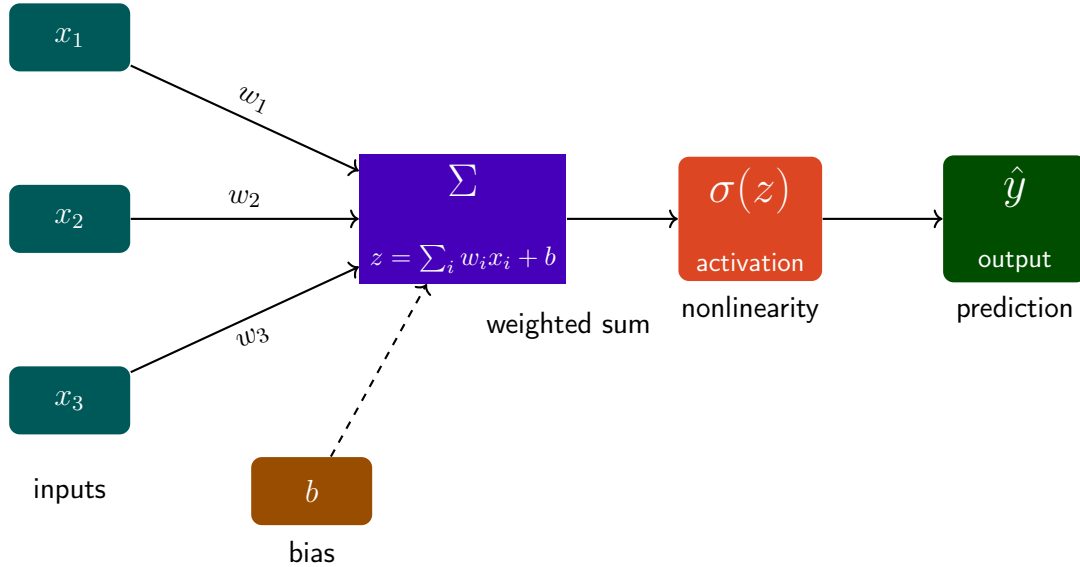


Figure 3.1: Schematic representation of a single neuron, showing the inputs x_1, x_2, x_3 , their weights, the bias term, the weighted sum z , the activation function $\sigma(z)$, and the resulting prediction $\hat{y}$.

Each element of this diagram is detailed below.

Inputs The inputs are single numbers, the network is evaluated at one point at a time, with x denoting position and t denoting time.

Weights Each input is multiplied by a weight, w_1 or w_2 , before entering the neuron. A weight expresses how much the neuron should trust that particular input: a weight near zero means the input is effectively ignored, while a large weight means it strongly influences the result. Weights are initialized randomly and adjusted during training.

Bias The bias b shifts the neuron's output up or down, independently of the inputs. Without a bias, a neuron computing $y = wx$ would always pass through the origin; adding a bias gives $y = wx + b$, which can be shifted to fit a line at any height.

Weighted sum The quantity z is the weighted sum of the inputs plus the bias,

$$z = w_1x + w_2t + b. \quad (3.1)$$

This expression is purely linear. Geometrically, it describes a flat plane in the space of inputs, and is not, by itself, powerful enough to represent the kind of functions a network must learn.

Activation function The function σ is applied to z immediately afterward, and is what introduces nonlinearity into the computation. If a network were built by stacking layers without any nonlinear activation, the entire composition, no matter how many layers it contains, would reduce mathematically to a single linear transformation. Stacking a hundred purely linear layers produces nothing more expressive than one linear layer. This is an issue since the wave function $\psi(x, t)$ is not a linear function, and the partial differential equation it satisfies is not linear either, the net-

work must be able to oscillate, bend, and curve. When sufficiently many nonlinear units are combined across a network, the resulting combination becomes capable of approximating an extremely wide class of functions, a result formalized by the universal approximation theorem, which states that a network with at least one hidden layer and a suitable nonlinear activation can approximate any continuous function to arbitrary accuracy, given enough neurons [41, 42].

The choice of which activation function to use is not arbitrary, and for the present work it is guided by a specific practical requirement. The loss function of the physics-informed neural networks developed later in this chapter involves first and second derivatives of the network's output with respect to its inputs, since these derivatives appear directly in the Schrödinger equation itself. This means the activation function must remain differentiable, with a well-defined and meaningful second derivative, everywhere in the domain. The rectified linear unit (ReLU), one of the most common activation functions in standard deep learning, fails this requirement because it has a sharp kink at zero, where it is not differentiable, and its second derivative is zero almost everywhere, which causes serious difficulties when an equation's residual depends on second-order derivatives of the network's output [43, 44]. For this reason, smooth activation functions are required instead. In the present work, the hyperbolic tangent and the sine function are the standard choices, both being smooth and infinitely differentiable across their entire domain [11, 45, 46]. Figure 3.2 compares these activation functions.

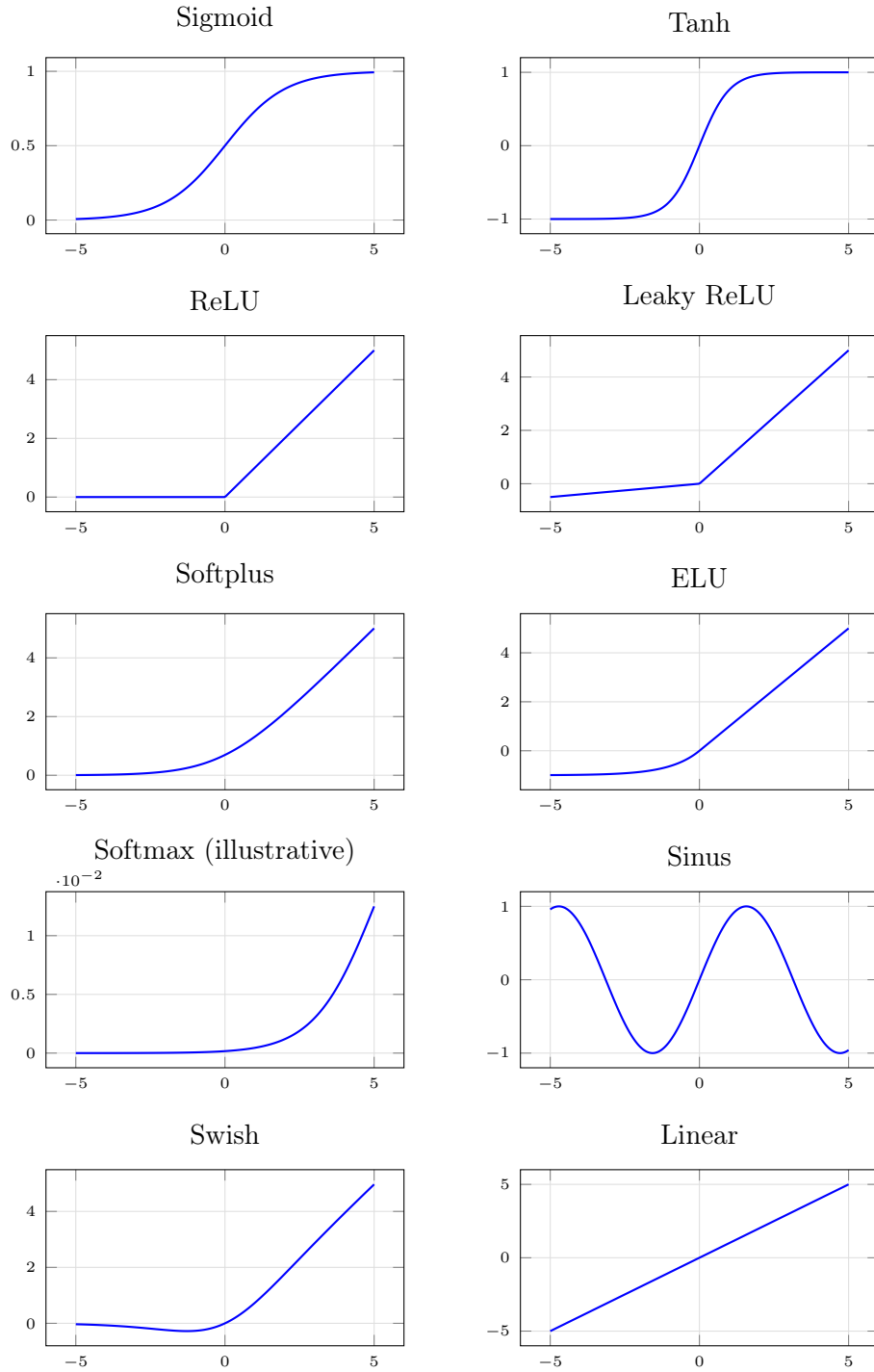


Figure 3.2: Comparison of common activation functions used in machine learning. ReLU and its variants are non-differentiable or have discontinuous second derivatives, making them unsuitable for physics-informed loss functions involving second-order derivatives. The hyperbolic tangent and sine are smooth everywhere and are the activation functions used in this work.

Output The output $\hat{y} = \sigma(z)$ is the neuron’s prediction. In the present work, the full network does not produce a single scalar output, but two values simultaneously: the real and imaginary parts of the wave function, $\text{Re}(\psi)$ and $\text{Im}(\psi)$, each obtained from its own output neuron operating on the same underlying hidden representation.

3.2.3 From One Neuron to a Deep Network

A single neuron, on its own, can only represent a very limited family of functions. The expressive power of a neural network comes from stacking many neurons into layers as mentioned earlier, and stacking several such layers one after another, so that the output of one layer becomes the input to the next [36].

The figure below shows this architecture for the present work. The input layer receives the position x and the time t . This is followed by several hidden layers, each containing several neurons, every one of which performs exactly the operation described in the previous section: a weighted sum of its inputs, plus a bias, passed through the activation function σ . The only difference is scale, the same simple operation, repeated many times, with different weights and biases at every neuron in every layer.

The output layer produces two values rather than one in our case, since the network is trained to predict both the real and imaginary parts of the wave function, $\text{Re}(\psi)$ and $\text{Im}(\psi)$. For example, if the true wave function at $x = 1$, $t = 0.5$ has the value $3 + 2i$, the network is trained so that its two output neurons return 3 and 2 respectively, which together reconstruct the complex value $\psi(1, 0.5) = 3 + 2i$.

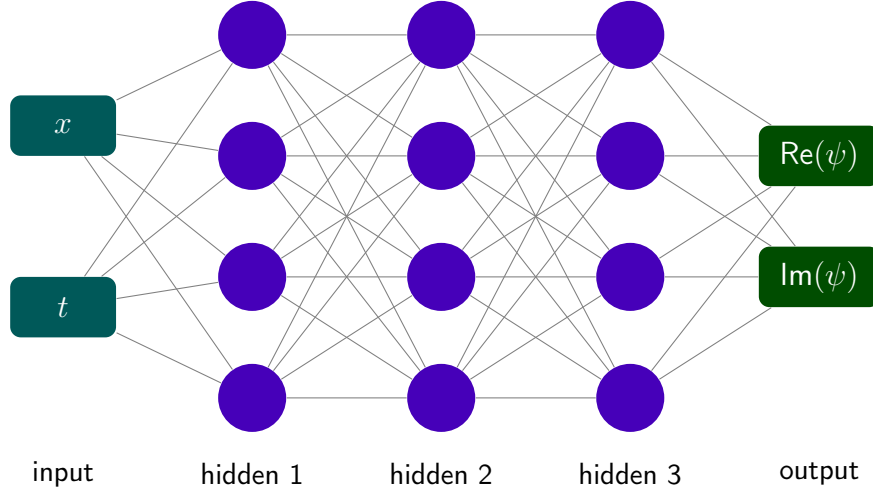


Figure 3.3: Schematic representation of a deep network. The input layer receives the position x and time t . Each hidden layer consists of several neurons, each performing the same operation: a weighted sum of its inputs, a bias, and an activation function.

What changes from one neuron to the next is only the specific weights and bias each one holds. It is this repetition of a simple operation, at scale, across many neurons and many layers, that allows the network as a whole to produce far more complex behavior than any single neuron could on its own.

3.2.4 Loss Function, Backpropagation, and Gradient Descent

Once the network produces an output, a way to measure how wrong that output is must be defined before any learning can take place.

Loss function The loss function takes the network's prediction $\hat{y}$ and the true value y and returns a single number quantifying the discrepancy between them. For a standard network trained purely on data, the most common choice is the mean squared error,

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^N |y_i - \hat{y}_i|^2, \quad (3.2)$$

where N is the number of training points. A large loss means the predictions are far from the true values, a loss of zero means they are perfect. The goal of training

is to make this quantity as small as possible.

Backpropagation Once the loss has been computed, the question becomes: which weights in the network should be adjusted, and by how much? Somewhere in the network there are weights that are making the loss large, but identifying which ones, and how strongly each one contributes to the error, requires tracing the loss backward through every layer.

Here we talk about the chain rule of calculus. The loss $\mathcal{L}$ depends on the network's output $\hat{y}$, which depends on the last layer's weights, which depend on the previous layer's outputs, which depend on their weights, and so on all the way back to the first layer. The dependency chain can be written as

$$\mathcal{L} \leftarrow \hat{y} \leftarrow w_3 \leftarrow h_2 \leftarrow w_2 \leftarrow h_1 \leftarrow w_1, \quad (3.3)$$

where h_1 and h_2 denote the outputs of hidden layers one and two respectively. To find how much the loss changes when the first weight w_1 is nudged, the chain rule gives

$$\frac{\partial \mathcal{L}}{\partial w_1} = \frac{\partial \mathcal{L}}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial h_2} \cdot \frac{\partial h_2}{\partial w_1}. \quad (3.4)$$

The interpretation is that changing w_1 changes h_2 , which changes $\hat{y}$, which changes the loss. The chain rule multiplies these small effects together to determine the total influence of w_1 on $\mathcal{L}$.

Backpropagation is the algorithm that applies this chain rule across every weight and every bias in the entire network, starting from the output and walking backward to the input [47]. In PyTorch, this is handled automatically via autograd. During the forward pass, PyTorch records every mathematical operation performed, and when backpropagation is requested, it walks backward through these recorded operations and computes every derivative at once, without the user writing a single line of calculus by hand [48, 38].

Gradient descent Once the derivative $\frac{\partial \mathcal{L}}{\partial w}$ has been computed for every weight, the optimizer uses these values to update each weight according to the rule [49]

$$w \leftarrow w - \alpha \frac{\partial \mathcal{L}}{\partial w}, \quad (3.5)$$

where α is the learning rate. As a concrete example, suppose the current weight is $w = 10$, the derivative is $\frac{\partial \mathcal{L}}{\partial w} = 5$, and the learning rate is $\alpha = 0.1$. The updated weight becomes $10 - 0.1 \times 5 = 9.5$. The network has determined that this weight was increasing the error, and has reduced it slightly. The derivative $\frac{\partial \mathcal{L}}{\partial w}$ indicates the direction in which the loss increases. Subtracting it therefore moves the weight in the direction that reduces the loss. The learning rate α controls the size of each step: too large, and the update overshoots the minimum; too small, and training converges extremely slowly.

These four operations together define the training loop, which repeats until the loss is sufficiently small.

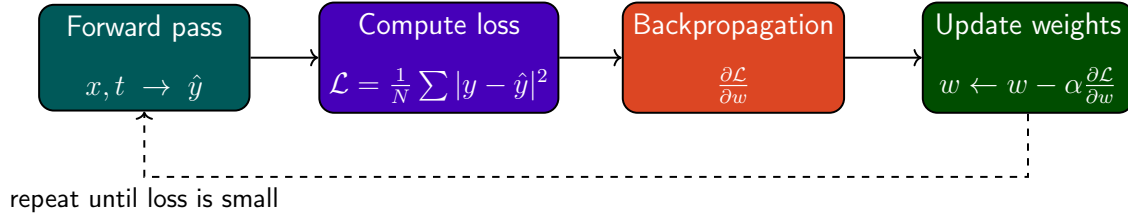


Figure 3.4: The training loop. A forward pass produces the prediction $\hat{y}$; the loss quantifies the error; backpropagation computes the derivative of the loss with respect to every weight; gradient descent updates the weights. This cycle repeats until the loss is sufficiently small.

3.3 Physics-Informed Neural Networks

3.3.1 From Data to Physics: Loss Function Changing

The standard neural network described in the previous section learns purely from labeled data. This requires a large amount of labeled examples, and for quantum

mechanical systems such large datasets of wave function measurements are rarely, if ever, available.

A Physics-Informed Neural Network (PINN) addresses this by changing what the loss measures [11, 10]. Instead of comparing the prediction only against labeled data, the network is made to obey the governing partial differential equation. The architecture is unchanged, the same network, the same neurons, the same layers, only the loss function is different. This loss is composed of three terms,

$$\mathcal{L} = \mathcal{L}_{\text{PDE}} + \lambda_1 \mathcal{L}_{\text{IC}} + \lambda_2 \mathcal{L}_{\text{BC}}. \quad (3.6)$$

As developed below, these three terms are not all of the same nature, one of them is a residual of a differential equation, while the other two are direct comparisons against known values.

In a standard neural network, the prediction is compared against a true labeled value, and gradient descent asks how each weight affects that error. In a physics-informed neural network, the prediction is compared against the requirement that the governing equation must equal zero, and gradient descent asks how each weight affects the violation of physics, rather than the distance to a labeled value.

	Standard NN	PINN
Compared against	true value y	$\text{PDE} = 0$
Loss	$\mathcal{L} = (y - \hat{y})^2$	$\mathcal{L} = R^2$
$\partial\mathcal{L}/\partial w$ tells us	effect on error	effect on physics violation

Table 3.1: Comparison between a standard neural network and a physics-informed neural network. The target in the PINN case is not a value but a condition: the residual must equal zero.

3.3.2 The PDE Residual

Consider the Schrödinger equation written in residual form,

$$R(x, t) = i\hbar \frac{\partial \psi}{\partial t} + \frac{\hbar^2}{2m} \frac{\partial^2 \psi}{\partial x^2} - V(x)\psi. \quad (3.7)$$

The term residual is used specifically here, and not for the conditions discussed in the next section, because $R(x, t)$ comes from a differential equation, it involves derivatives of ψ , and measures whether these derivatives, once computed, balance according to the equation. If $\psi(x, t)$ is an exact solution, every term in Equation 3.7 cancels exactly and $R(x, t) = 0$ everywhere. If $\psi(x, t)$ is not a true solution, the terms no longer cancel, and the leftover imbalance is precisely the residual. Crucially, this check requires no separately known target value for ψ itself, ψ is only compared against its own derivatives, through the equation, not against an external answer.

This is the same logic as a force balance in Newtonian mechanics, written as a sum of forces equal to zero rather than equal to some other value, a nonzero result signals an imbalance, something left over that should not be there. The Schrödinger equation is not a quantity to be approximated or fit to data, it is a constraint that any true physical solution must satisfy exactly, with nothing left over.

The residual cannot be checked at every point of a continuous domain. Instead, a set of points (x, t) is sampled randomly across the domain, with both coordinates free, called collocation points [11, 50]. These points are not learned and carry no special meaning on their own, they are simply locations at which the residual is evaluated and penalized. At each collocation point, the network's output $\psi_\theta(x, t)$ is computed, its derivatives with respect to t and x are obtained automatically via autograd [48], and these are substituted into Equation 3.7 to obtain $R(x, t)$. The corresponding loss term is the mean squared residual over all collocation points,

$$\mathcal{L}_{\text{PDE}} = \frac{1}{N_{\text{PDE}}} \sum_{i=1}^{N_{\text{PDE}}} |R(x_i, t_i)|^2. \quad (3.8)$$

3.3.3 Initial and Boundary Condition Losses

The PDE residual alone does not determine a unique solution, the same differential equation admits infinitely many solutions unless further constraints are imposed [11]. These constraints are the initial condition and the boundary conditions, and unlike the PDE term, they are not differential equations, there are no derivatives to balance, only a direct requirement that the network's output match a known value at specific points. Enforcing them is correspondingly simpler, an ordinary squared error, of exactly the kind used in a standard neural network, but applied only at points deliberately chosen to lie at $t = 0$ or at the spatial boundary.

This is an important distinction from the collocation points used for the PDE loss. Collocation points are sampled freely, with both x and t random. Initial condition points are constructed by sampling x randomly but fixing $t = 0$ for every point in the batch. Boundary condition points are constructed by sampling t randomly but fixing x at the domain edge, $x = a$ or $x = b$, for every point in the batch. In all three cases, the points are generated deliberately to land where they are needed, not discovered by chance among a single pool of random samples.

The initial condition loss is evaluated at these $(x, 0)$ points, where the true initial wave function $\psi_0(x)$ is known and supplied by the user as part of the problem setup.

$$\mathcal{L}_{\text{IC}} = \frac{1}{N_{\text{IC}}} \sum_{i=1}^{N_{\text{IC}}} |\psi(x_i, 0) - \psi_0(x_i)|^2. \quad (3.9)$$

The boundary condition loss is evaluated at the (a, t) and (b, t) points, enforcing that the wave function takes its prescribed boundary value, here zero, for a particle confined to a finite region,

$$\mathcal{L}_{\text{BC}} = \frac{1}{N_{\text{BC}}} \sum_{i=1}^{N_{\text{BC}}} (|\psi(a, t_i)|^2 + |\psi(b, t_i)|^2). \quad (3.10)$$

The network itself receives no special information about which type of point it is being given there is only one network, and it only ever takes (x, t) as input. What

differs is how each batch of points is constructed beforehand, and how its output is used afterward to compute the loss. Only the PDE term is a residual in the strict sense; the initial and boundary condition terms are ordinary discrepancies from a known target.

3.3.4 Joint Training on All Three Losses

A point that deserves particular emphasis is that the three losses are not learned one after another, in sequence, but jointly, within every single training step [11]. At each iteration, three batches of points are generated, the collocation batch, the initial condition batch, and the boundary condition batch, typically several hundred or a few thousand points each. All three batches are passed through the same network in that same step, producing $\mathcal{L}_{\text{PDE}}$, $\mathcal{L}_{\text{IC}}$, and $\mathcal{L}_{\text{BC}}$, which are then summed into the single total loss. Backpropagation is performed on this combined value, and a single weight update follows [47]. The network is therefore never permitted to satisfy one condition first and address the others afterward but from the very first training step, every weight update reflects a balance between reducing the physics violation, matching the initial condition, and matching the boundary condition simultaneously. Over many such steps, the network converges toward a configuration of weights that keeps all three contributions small at once, rather than minimizing them in isolation.

3.4 Inverse Physics-Informed Neural Networks

3.4.1 Unknown Parameters

In the forward problem developed above, every term in the Schrödinger equation is known, the potential $V(x)$ is given, and the network’s task is to find the wave function $\psi(x, t)$ that satisfies the equation together with the initial and boundary conditions. The inverse problem reverses this logic [11]. The potential, or more generally some physical parameter appearing inside the governing equation, is not

known in advance. What is available instead is a set of observations of the wave function itself, measurements of ψ at certain points (x, t) , and the question becomes: what value of the unknown parameter is consistent with both the governing equation and these observations?

The network still takes (x, t) as input and outputs the wave function, exactly as in the forward case, and the architecture itself is unchanged. What changes is what is being learned. In the forward problem, only the weights and biases of the network are trainable parameters. In the inverse problem, the unknown physical parameter, denoted α here for generality (this may represent the potential $V(x)$, or more generally any unknown coefficient appearing in the governing equation), is trained alongside the weights, rather than supplied as a fixed input.

3.4.2 The Data Loss Term

The loss function retains the same PDE, initial condition, and boundary condition terms developed previously, but a fourth term is added, penalizing disagreement between the network's predictions and the available observations [11, 51]. Suppose a set of measurements $u_{\text{obs}}(x_i, t_i)$ is available at certain points (x_i, t_i) . The data loss compares the network's prediction at these same points against the measured values,

$$\mathcal{L}_{\text{data}} = \frac{1}{N_{\text{data}}} \sum_{i=1}^{N_{\text{data}}} |u_{\theta}(x_i, t_i) - u_{\text{obs}}(x_i, t_i)|^2. \quad (3.11)$$

Like the initial and boundary condition losses, and unlike the PDE term, this is an ordinary discrepancy from a known target, not a residual, there is no equation being checked here, only a direct comparison between a prediction and a measurement.

The total loss for the inverse problem is therefore

$$\mathcal{L} = \mathcal{L}_{\text{PDE}} + \mathcal{L}_{\text{IC}} + \mathcal{L}_{\text{BC}} + \mathcal{L}_{\text{data}}. \quad (3.12)$$

3.4.3 Two Gradients

In the forward problem, only one derivative is needed for each weight, $\partial\mathcal{L}/\partial w$. In the inverse problem, a second derivative is needed as well, $\partial\mathcal{L}/\partial\alpha$, expressing how the total loss depends on the unknown physical parameter itself. Both are obtained from the same backpropagation pass [47, 48], since α enters the computational graph exactly as the weights do, through the PDE residual term in which it appears.

Training therefore updates two distinct kinds of quantities at every step,

$$w \leftarrow w - \mu \frac{\partial\mathcal{L}}{\partial w}, \quad \alpha \leftarrow \alpha - \mu \frac{\partial\mathcal{L}}{\partial\alpha}, \quad (3.13)$$

where μ is the learning rate. The first rule adjusts the network's weights, exactly as before, to better satisfy the combined loss. The second rule adjusts the unknown parameter itself, nudging it toward the value that makes the network's predictions more consistent with the governing equation and the observed data simultaneously.

In practice, this distinction is reflected directly in how the parameter is defined in code. In the forward problem, the parameter is known and is simply assigned a fixed numerical value before training begins, it never changes. In the inverse problem, the parameter is instead declared as a trainable quantity, in the same way the network's own weights are declared trainable, so that the optimizer updates it at every step along with the rest of the network.

3.4.4 Output of Training

At the end of training, an inverse PINN does not produce only an approximation of the wave function, as in the forward case. It produces two results simultaneously, the learned wave function $\psi_\theta(x, t)$, and the learned value of the previously unknown

parameter α . This value is not guaranteed to be exact, but it is the value that, jointly with the learned weights, best reconciles the governing equation, the initial and boundary conditions, and the observed data, all enforced together through the same combined loss [11].

The full pipeline of the inverse PINN, including both backpropagation paths, is summarized schematically in Figure 3.5.

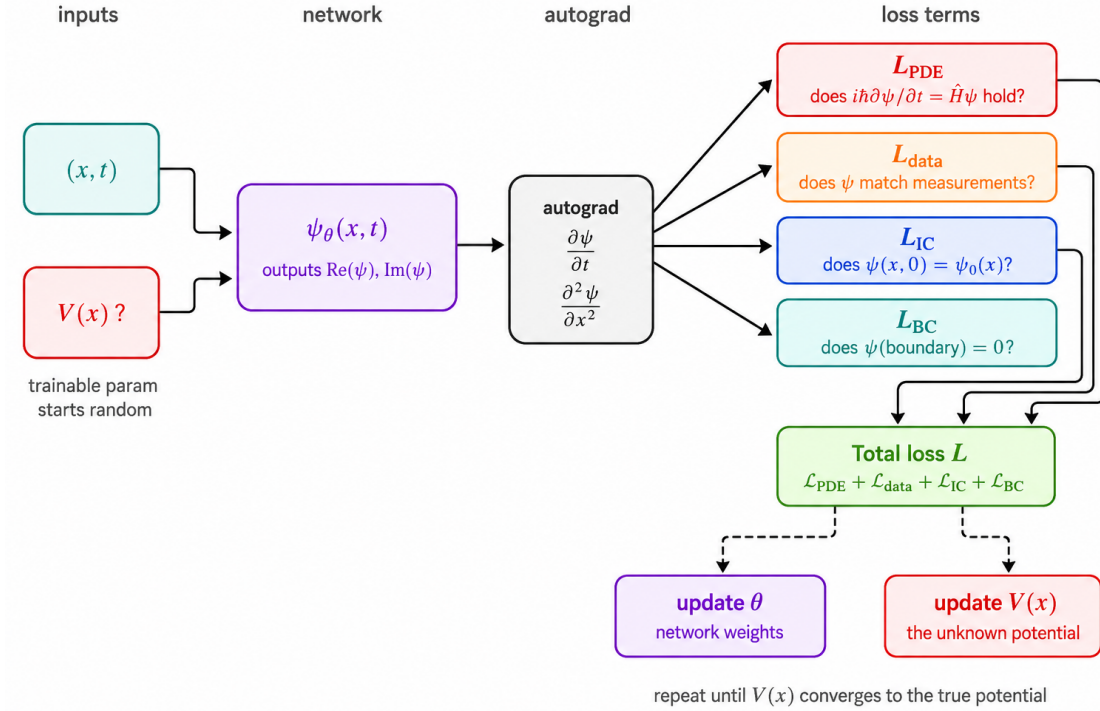


Figure 3.5: Pipeline of the inverse PINN. The inputs (x, t) and the trainable potential $V(x)$ are passed through the network to obtain $\psi_\theta(x, t)$; autograd computes the required derivatives, feeding into the PDE, data, initial condition, and boundary condition losses. These are summed into the total loss, and backpropagation updates both the network weights θ and the unknown potential $V(x)$ simultaneously.

3.4.5 Non-Uniqueness in the Inverse Problem

A difficulty already encountered in the classical setting, discussed in Chapter 2 for Tikhonov regularization, reappears here and deserves emphasis [8, 23]. The inverse problem may not have a unique solution. Two different potentials, $V_1(x)$ and $V_2(x)$, can produce wave functions that are nearly indistinguishable from one

another. If the available measurements are noisy or incomplete, the data loss can be equally small for both, and the network has no mechanism to prefer one over the other. Training simply finds a minimum of the loss landscape, not necessarily the minimum corresponding to the true potential.

Figure 3.6 illustrates this. Without further constraints, the loss landscape with respect to $V(x)$ may contain several minima of comparable depth, one of which corresponds to the true potential and others to incorrect ones that nonetheless reproduce the observed data closely. With additional information, whether more data or appropriate regularization, the landscape can be reshaped so that a single minimum remains.

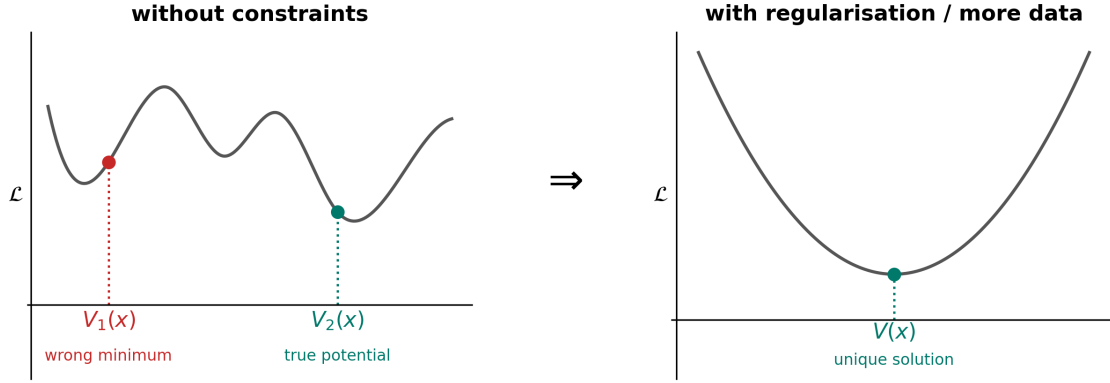


Figure 3.6: Loss landscape with respect to the unknown potential $V(x)$. Left: without constraints, two distinct potentials $V_1(x)$ and $V_2(x)$ produce similarly small loss, since both are consistent with the available data. Right: with regularisation or additional data, the ambiguity is removed and a single minimum corresponds to the true potential.

Several strategies can address this ambiguity. The most direct is simply to use more data: with measurements at more points, in more conditions, fewer potentials remain consistent with all of them simultaneously. A second strategy is to add a regularization term to the loss, exactly as in the Tikhonov approach of Chapter 2, penalizing potentials that are unnecessarily wild or non-smooth, and biasing the solution toward the simplest function consistent with the data [8, 9]. A third strategy is to incorporate physical prior knowledge directly: if it is known in advance that

the potential must be symmetric, positive, or of a particular functional form, this information can be built into the network or the loss, eliminating entire families of incorrect solutions before training even begins.

It is worth stating plainly that non-uniqueness is not a weakness specific to physics-informed neural networks. It is a property of the inverse problem itself, and Tikhonov regularization faces exactly the same difficulty [23]. This has recently been given a rigorous mathematical foundation: Shumaylov et al. have shown that for non-chaotic systems, trajectory data alone is provably insufficient to uniquely identify the governing dynamics, and that physical priors are a mathematical necessity rather than a computational convenience [52]. The harmonic oscillator, being non-chaotic, is a canonical example of this phenomenon: its periodic wave function traces the same path indefinitely, and that path is consistent with infinitely many potentials in the absence of additional constraints. What the PINN loss provides, and what Tikhonov regularization also provides in its own way is precisely the physical structure needed to restore uniqueness. What distinguishes the PINN approach is how this structure is incorporated. Regularization and physical constraints enter directly into a single continuous loss, enforced everywhere in the domain, rather than only at the discrete points of a grid.

3.5 Summary

This chapter developed Physics-Informed Neural Networks from first principles [11, 12]. The shift from rule-based to data-driven methods, the construction and training of a standard neural network, and the extension of this framework to enforce the Schrödinger equation directly through the loss function rather than through a mesh.

For the forward problem, the PDE residual was distinguished from the initial and boundary condition losses, the former a genuine measure of physical violation,

the latter ordinary discrepancies from known values, all three trained jointly within a single combined loss. For the inverse problem, this framework was extended by adding a data loss and declaring the unknown potential $V(x)$ as a trainable parameter alongside the network's weights, with both updated simultaneously through backpropagation. The resulting non-uniqueness was shown to be a property of the inverse problem itself, not of the PINN approach, addressable through additional data, regularization, or physical priors.

Chapters 2 and 3 together establish two complete frameworks, classical and physics-informed, for the forward and inverse Schrödinger problems on the quantum harmonic oscillator. The following chapter turns to their implementation and comparison: Crank-Nicolson against the forward PINN, and Tikhonov regularization against the inverse PINN.

Chapter 4

Implementation of Forward Solvers and Results

4.1 Physical System, Theoretical Background and Analytical Solution

4.1.1 Physical System

The system under study is 1D quantum harmonic oscillator, $V(x) = \frac{1}{2}x^2$, governed by the time-dependent Schrödinger equation with $\hbar = m = \omega = 1$ in reduced atomic units. Although the true harmonic oscillator is defined on $(-\infty, +\infty)$, the domain is truncated to $[-3, 3]$ with Dirichlet boundary conditions $\psi(\pm 3, t) = 0$. Other types of boundary conditions could in principle be imposed at the truncation walls: Neumann conditions prescribe the derivative $\partial_x \psi$ at the boundary, while Robin (or Fourier) conditions prescribe a linear combination of the function and its derivative, $\alpha\psi + \beta \partial_x \psi = g$. These are the natural choice when the physics imposes a flux or a radiation condition at the boundary. Here, however, the harmonic potential confines the wave function exponentially: for the initial state and time window considered, both ψ and $\partial_x \psi$ are negligible near $x = \pm 3$, so the homogeneous Dirich-

let condition $\psi(\pm 3, t) = 0$ is the simplest condition consistent with the physics. It is standard practice in computational quantum mechanics, introduces no physical artifact for a confined state, and is also the easiest to enforce exactly in both the Crank–Nicolson scheme and the PINN formulation. All simulation parameters are summarized in Table 4.1.

Table 4.1: Forward problem test case parameters.

Quantity	Symbol	Value
Spatial domain	$[x_{\min}, x_{\max}]$	$[-3, 3]$
Spatial points	N_x	2001
Spatial step	Δx	$\approx 3.0 \times 10^{-3}$
Time domain	$[t_{\min}, t_{\max}]$	$[0, 0.2]$
Time steps	N_t	201
Time step	Δt	10^{-3}
Physical constants	$\hbar, m, \omega$	1 (reduced atomic units)
Potential	$V(x)$	$\frac{1}{2}x^2$
Initial condition	$\psi(x, 0)$	$C \sin(\pi x), C = \sqrt{2}$
Boundary conditions	$\psi(\pm 3, t)$	0 (Dirichlet)

The initial state

$$\psi(x, 0) = C \sin(\pi x), \quad C = \sqrt{2}, \quad (4.1)$$

vanishes at both walls by construction. This choice is motivated by several considerations. First, since $\sin(\pi x)$ is an odd function, it decomposes exclusively onto the odd harmonic oscillator eigenstates $n = 1, 3, 5, \dots$ by Fourier–Hermite projection, so the exact time evolution is available analytically as a rapidly converging

eigenstate expansion. This provides a pointwise reference solution against which the Crank–Nicolson and PINN solvers can be quantitatively benchmarked. Second, because the state is not a single eigenstate but a superposition, the resulting dynamics are genuinely time-dependent: the probability density evolves nontrivially over the time window, which makes the test case more demanding than a stationary state whose density would remain static. Third, the sinusoidal profile vanishes exactly at $x = \pm 3$, so the initial condition is automatically compatible with the Dirichlet boundary conditions, avoiding any inconsistency between the initial and boundary data at the corners of the space–time domain. Finally, its amplitude is concentrated well inside the truncated domain, consistent with the assumption that the walls introduce no physical artifact.

Each method is evaluated on four metrics computed over the full trajectory:

1. L^2 **error** against the eigenvalue reference:

$$\varepsilon_{L^2}(t) = \|\psi_{\text{method}}(\cdot, t) - \psi_{\text{ref}}(\cdot, t)\|_{L^2}, \quad (4.2)$$

and its temporal mean $\bar{\varepsilon}_{L^2} = \frac{1}{N_t} \sum_{k=0}^{N_t-1} \varepsilon_{L^2}(t_k)$.

2. **Norm drift**: $\delta N(t) = \|\psi(\cdot, t)\|_{L^2} - 1$, measuring violation of probability conservation.

3. **Density error**:

$$\varepsilon_\rho(t) = \left\| |\psi_{\text{method}}(\cdot, t)|^2 - |\psi_{\text{ref}}(\cdot, t)|^2 \right\|_{L^2}, \quad (4.3)$$

phase-insensitive and directly tied to the physical observable.

4. **Total computation time** in seconds, measured on an Intel Core i9-14900KF processor with NVIDIA RTX A4000 GPU.

Results are presented through density snapshots $|\psi(x, t)|^2$ at selected time in-

stants, L2 error curves over time, norm conservation plots, and computation time comparisons.

The eigenvalue method serves as the discrete reference. The analytical solution is also shown for context, but its gap with the numerical methods is physical which is a consequence of the box-versus-free-space difference.

4.1.2 Theoretical Background

Starting from the time-dependent Schrödinger equation with the harmonic potential injected in it, we apply separation of variables by writing $\psi(x, t) = \varphi(x) T(t)$. The temporal part gives immediately $T(t) = e^{-iEt/\hbar}$ for each energy level E .

For the spatial part, the potential grows unboundedly as $x \rightarrow \pm\infty$, so the wave function must decay to zero at large $|x|$. Functions like sin or cos oscillate forever and do not decay therefore they are physically unacceptable. A Gaussian $e^{-\alpha^2 x^2/2}$ does decay, so physicists use it as an ansatz: the spatial solution is assumed to take the form $\varphi(x) = e^{-\alpha^2 x^2/2} u(x)$, where $u(x)$ is an unknown function to be determined.

Substituting into the spatial Schrödinger equation and simplifying, the equation for $u(x)$ becomes the Hermite differential equation, whose physically acceptable solutions are the Hermite polynomials $H_n(x)$. These are simply polynomials, $H_0 = 1$, $H_1 = 2x$, $H_2 = 4x^2 - 2$, and so on, that produce the n nodes of the n -th excited state. The Gaussian alone gives a smooth bell curve with no nodes; multiplying by H_n adds the correct number of oscillations on top of it.

The complete normalized eigenstate is:

$$\varphi_n(x) = \left(\frac{\alpha}{\pi}\right)^{1/4} \frac{1}{\sqrt{2^n n!}} H_n(\alpha x) e^{-\alpha^2 x^2/2}, \quad \alpha = \sqrt{\frac{m\omega}{\hbar}}, \quad (4.4)$$

which is the product of a normalization constant, the Hermite polynomial shape, and the Gaussian envelope. Each eigenstate carries a quantized energy $E_n = (n + \frac{1}{2})\hbar\omega$.

Since the initial state $\psi(x, 0) = \sqrt{2}\sin(\pi x)$ is not a single eigenstate, it is de-

composed over the eigenbasis:

$$\psi(x, t) = \sum_{n=0}^{+\infty} c_n \varphi_n(x) e^{-iE_n t/\hbar}, \quad (4.5)$$

where each mode evolves independently with its own phase factor, and the projection coefficients $c_n = \langle \varphi_n | \psi_0 \rangle$ are computed numerically by the trapezoidal rule:

$$c_n \approx \Delta x \sum_{j=0}^{N_x-1} \varphi_n^*(x_j) \psi_0(x_j). \quad (4.6)$$

4.1.3 Analytical Solution Implementation

The simulation relies on dedicated components handling the spatial and temporal grids, the harmonic potential, the discrete Hamiltonian, and the initial conditions. The initial state used throughout all forward experiments is $\psi(x, 0) = \sqrt{2} \sin(\pi x)$, normalized to unit L^2 norm on $[-3, 3]$.

The analytical solution proceeds in four steps:

1. **Eigenstates** each harmonic oscillator eigenfunction $\varphi_n(x)$ is evaluated on the spatial grid according to equation (4.4), a Gaussian envelope with n oscillations on top. All eigenstates are stacked into a matrix.
2. **Projection** the initial condition $\psi(x, 0)$ is decomposed onto the eigenbasis by computing the inner products $c_n = \langle \varphi_n | \psi_0 \rangle$ via the trapezoidal rule (equation (4.6)). Only odd modes $n = 1, 3, 5, \dots$ contribute by symmetry of the initial condition.
3. **Time evolution** each mode is multiplied by its phase factor $e^{-iE_n t/\hbar}$ independently. Each mode rotates in the complex plane at its own frequency $E_n/\hbar$; their superposition gives the full wave function at any time t .
4. **Output** the full 2001×201 complex matrix $\psi(x, t)$ is returned. The analytical solution evaluates the known formula directly, it does not solve any equation

numerically.

This solution serves as the continuous reference. Its gap with the discrete numerical methods reflects the $\mathcal{O}(\Delta x^2)$ spatial discretization error of the finite-difference Hamiltonian, a physical difference, not a numerical failure, discussed further later on.

4.2 Eigenvalue Method

4.2.1 Implementation

The eigenvalue method builds the Hamiltonian as a tridiagonal matrix over the $N_x - 2 = 1999$ interior points, using second-order centred finite differences:

$$H_{jk} = \begin{cases} -\frac{\hbar^2}{m\Delta x^2} + V(x_j) & \text{if } k = j, \\ +\frac{\hbar^2}{2m\Delta x^2} & \text{if } k = j \pm 1, \\ 0 & \text{otherwise.} \end{cases} \quad (4.7)$$

This matrix is then diagonalized numerically to obtain all 1999 eigenvalues and eigenvectors. The initial condition is decomposed onto this eigenbasis by computing the projection coefficients $c_n = \langle \varphi_n | \psi_0 \rangle$, and each eigenstate is propagated independently via its exact phase factor $e^{-iE_n t/\hbar}$. The full wave function at any time t is recovered by superposition. Only eigenstates with non-negligible coefficients are retained, and the time evolution is computed in memory-efficient blocks.

4.2.2 Accuracy and Norm Conservation

Table 4.2 compares the five lowest discrete eigenvalues against the exact analytical levels $E_n = (n + \frac{1}{2})\hbar\omega$. The relative error grows with n , consistent with the $\mathcal{O}(\Delta x^2)$ truncation error of the finite-difference Hamiltonian.

Table 4.2: First five discrete eigenvalues $\tilde{E}_n$ compared to the analytical levels $E_n = (n + \frac{1}{2})\hbar\omega$.

n	E_n (analytical)	$\tilde{E}_n$ (discrete)	$ \tilde{E}_n - E_n /E_n$
0	0.5000	0.5001	2.0×10^{-4}
1	1.5000	1.5003	2.0×10^{-4}
2	2.5000	2.5007	2.8×10^{-4}
3	3.5000	3.5014	4.0×10^{-4}
4	4.5000	4.5024	5.3×10^{-4}

Norm conservation is exact by construction: time evolution amounts to evaluating a closed-form sum of phase factors, so the norm cannot drift. The measured maximum drift is $\delta N < 1.2 \times 10^{-14}$, pure floating-point arithmetic.

Figure 4.2 shows the probability density $|\psi(x, t)|^2$ over the full simulation window, and Figure 4.1 shows snapshots at selected time instants. The smooth, structured pattern reflects coherent superposition of odd harmonic eigenmodes and serves as the discrete reference for all subsequent comparisons.

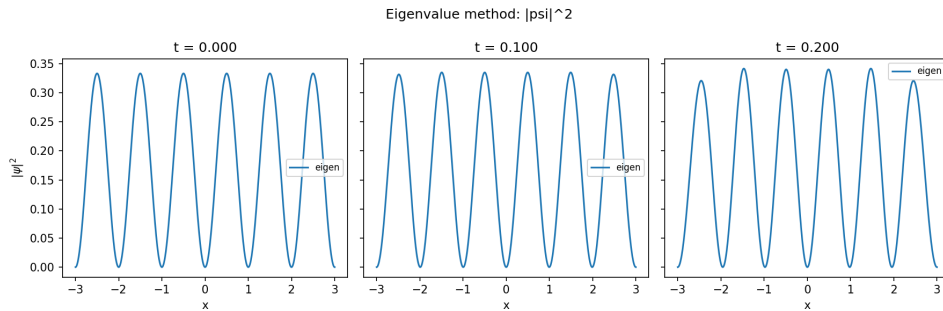


Figure 4.1: Snapshots of $|\psi(x, t)|^2$ at three time instants.

The three snapshots show that the wave function maintains its oscillatory shape across all times: the peaks remain at the same height (≈ 0.33), the zeros stay fixed at the walls, and while the pattern shifts slightly between snapshots, the overall

structure is fully preserved. There is no dissipation, no amplitude growth, and no deformation, the eigenvalue method conserves the shape and energy content of the wave function exactly.

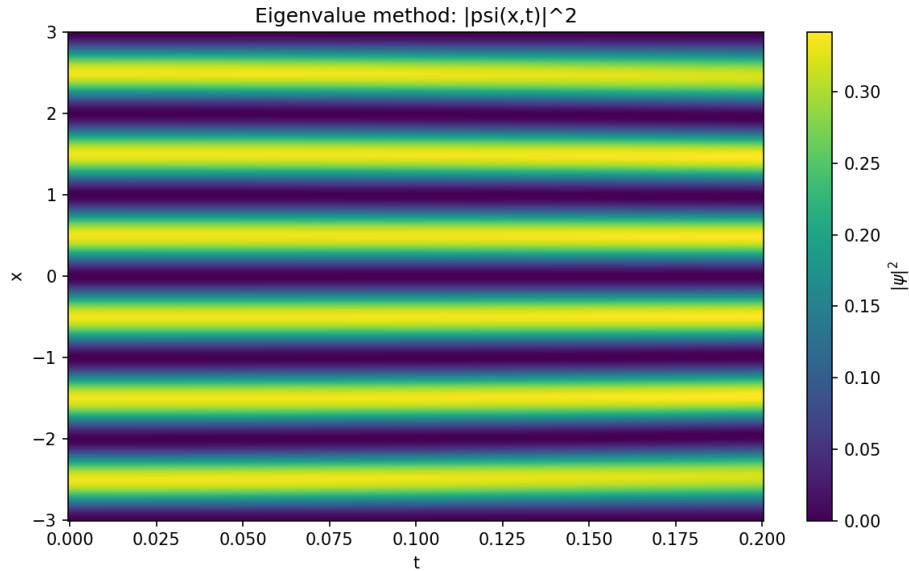


Figure 4.2: Full space-time evolution of $|\psi(x, t)|^2$ over $t \in [0, 0.2]$.

The heatmap reveals the complete space-time structure of the solution. The alternating bands of high density (yellow) and near-zero density (purple) shift smoothly in space as time progresses, reflecting the periodic oscillation of the wave function. Crucially, the bands show no distortion, fading, or drift over time, the pattern remains perfectly regular throughout the simulation window. If probability were leaking away, the yellow bands would dim progressively; the fact that they remain uniformly bright confirms that the norm is conserved to machine precision.

4.2.3 Computational Cost

Diagonalizing the 1999×1999 real symmetric tridiagonal matrix scales as $\mathcal{O}(N_x^3)$, which becomes prohibitive for large grids ($N_x > 10^4$) or higher dimensions. Once the eigenbasis is computed, temporal evaluation depends only linearly on N_t . Exact runtimes are reported in the global comparison later on.

4.3 Crank-Nicolson Method

4.3.1 Implementation

Among the many time-stepping schemes available, the Crank–Nicolson method is particularly well suited to the time-dependent Schrödinger equation. Because of the imaginary factor i in the equation, the eigenvalues of the evolution operator lie on the unit circle rather than inside it, and simple explicit schemes such as forward Euler are unconditionally unstable for this problem. The Crank–Nicolson scheme, in contrast, is unconditionally stable and, more importantly, unitary: it preserves the L^2 norm of the wave function exactly at the discrete level, which is essential for respecting the probabilistic interpretation over the time integration. It is also second-order accurate in both space and time. Alternative implicit schemes such as the Peaceman–Rachford alternating direction implicit (ADI) method share these stability properties, but their advantage lies in higher-dimensional problems, where they split a large multidimensional linear system into a sequence of inexpensive one-dimensional solves. For the one-dimensional system considered here, the Crank–Nicolson linear system is already tridiagonal and can be solved directly in $O(N_x)$ operations per time step, so no such splitting is needed.

The Crank-Nicolson scheme advances the wave function one time step Δt at a time by solving the linear system:

$$\underbrace{\left(I + \frac{i\Delta t}{2\hbar}H\right)}_A \psi^{n+1} = \underbrace{\left(I - \frac{i\Delta t}{2\hbar}H\right)}_B \psi^n, \quad (4.8)$$

where H is the discrete Hamiltonian from equation (4.7). This is the Cayley scheme — half implicit, half explicit — which is unconditionally stable for any time step Δt and preserves the norm of ψ exactly.

The key computational insight is that matrix A never changes throughout the simulation, it depends only on H , Δt , and $\hbar$, all of which are fixed. It is therefore

LU-factorized once before the time loop starts. At each of the 200 subsequent time steps, solving $A\psi^{n+1} = B\psi^n$ reduces to a cheap triangular solve using the precomputed factorization. The wall points are excluded from the system and kept fixed at zero, enforcing the Dirichlet boundary conditions exactly. The output is the full 2001×201 complex matrix $\psi(x, t)$.

4.3.2 Norm Conservation and Unitarity

The discrete evolution operator

$$U_{\Delta t} = \left(I + \frac{i\Delta t}{2\hbar} H \right)^{-1} \left(I - \frac{i\Delta t}{2\hbar} H \right) \quad (4.9)$$

is unitary for any $\Delta t > 0$, since its spectral amplification factor satisfies:

$$\left| \frac{1 - i\alpha\lambda}{1 + i\alpha\lambda} \right| = 1 \quad \forall \lambda \in \mathbb{R}. \quad (4.10)$$

This guarantees exact norm conservation at every time step by construction. The measured norm drift over the full trajectory is $\delta N = 9.3 \times 10^{-15}$, limited only by double-precision floating-point arithmetic.

4.3.3 Accuracy

The L^2 error against the eigenvalue reference grows monotonically from machine precision at $t = 0$ to $\varepsilon_{L^2}(T) \approx 9.2 \times 10^{-6}$ at $t = 0.2$, consistent with $\mathcal{O}(\Delta t^2)$ phase error accumulation. The density error ε_ρ is systematically lower than the complex L^2 error, confirming that the error is predominantly a phase shift rather than an amplitude error.

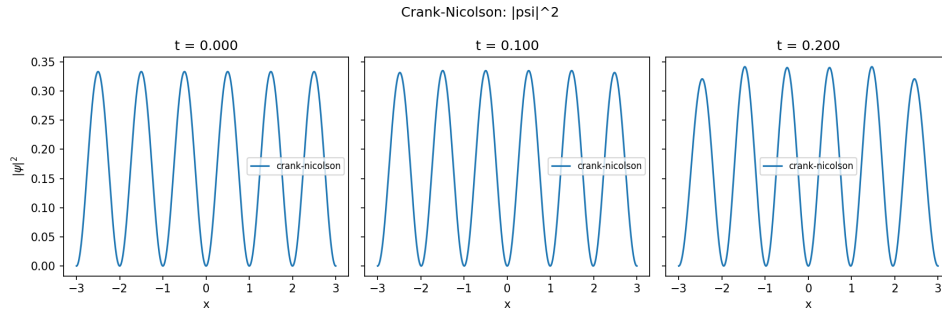


Figure 4.3: Snapshots of $|\psi(x, t)|^2$ at three time instants, computed by the Crank-Nicolson scheme.

The three snapshots show a fully preserved oscillatory structure with no dissipation or amplitude change across time steps. The peaks remain stable and the zeros stay fixed at the walls, visually confirming norm conservation.

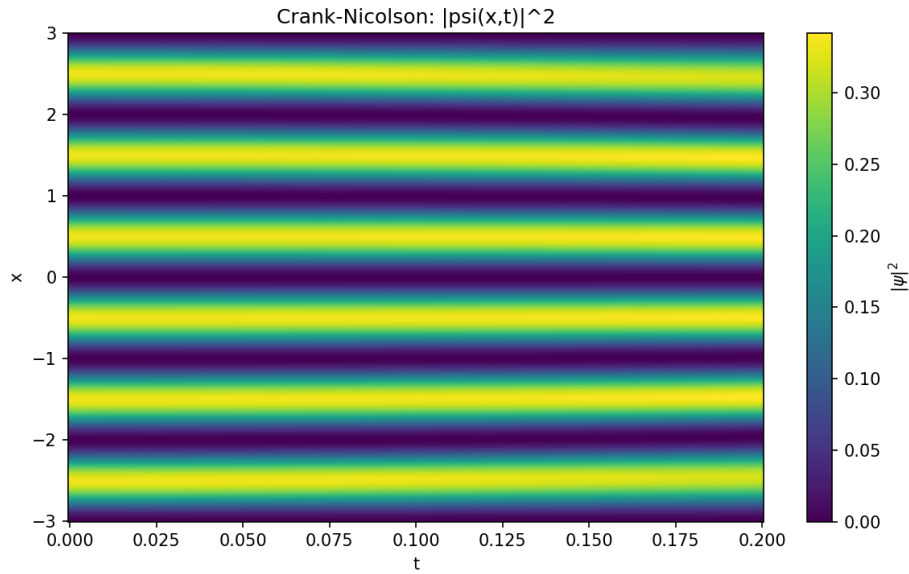


Figure 4.4: Full space-time evolution of $|\psi(x, t)|^2$ computed by the Crank-Nicolson scheme over $t \in [0, 0.2]$.

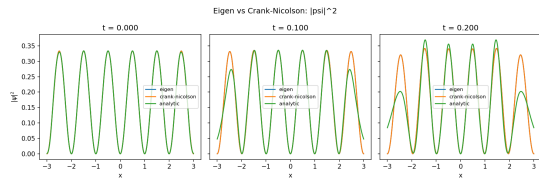
The heatmap shows regular, undistorted bands with no fading or drift throughout the simulation window, confirming that the scheme propagates the wave function faithfully without any loss of probability.

4.3.4 Computational Cost

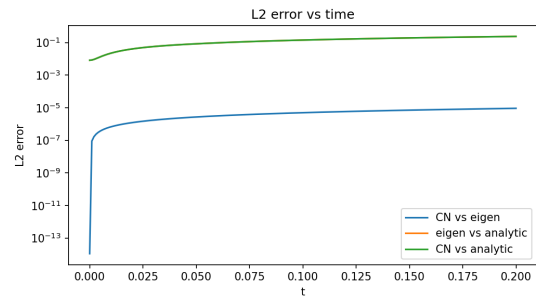
The LU factorization of the 1999×1999 sparse matrix A is performed once before the time loop. Each subsequent time step requires only a triangular solve in $\mathcal{O}(N_x)$, giving a total cost of $\mathcal{O}(N_x N_t)$.

4.4 Test 1: Eigenvalue Method vs Crank-Nicolson

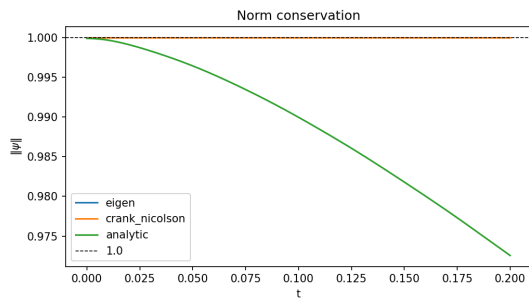
This first comparison establishes the classical baseline before introducing PINNs. Both methods solve the same box problem and are compared directly against each other and against the analytical free-space solution.



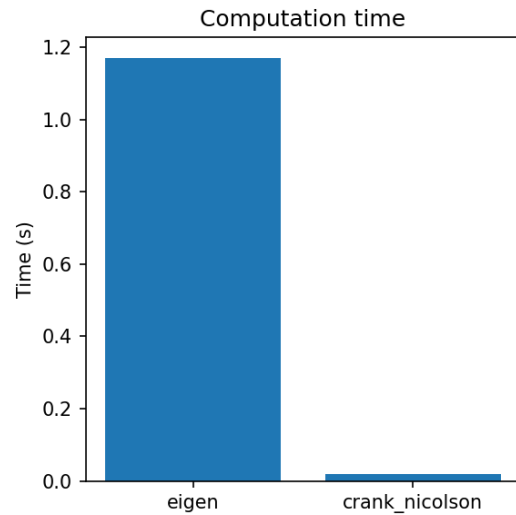
(a) Probability density $|\psi|^2$ Eigen vs Crank-Nicolson



(b) L^2 error



(c) Norm conservation



(d) Timings

Figure 4.5: Test 1 — Eigenvalue method vs Crank-Nicolson at $T = 0.2$.

Wave function evolution At $t = 0$, all three solutions overlap exactly. At $t = 0.1$ and $t = 0.2$, the eigenvalue method and Crank-Nicolson remain visually identical, while the analytical solution diverges near the walls. This is the box-versus-free-space effect: the analytical eigenstates are defined on the full real line and probability leaks past the walls, while both numerical methods enforce $\psi(\pm 3, t) = 0$ exactly.

Accuracy The L^2 error between Crank-Nicolson and the eigenvalue reference starts at 1.09×10^{-14} at $t = 0$ and grows monotonically to 9.2×10^{-6} at $t = 0.2$, four orders of magnitude below anything physically meaningful. The errors of both solvers against the analytical solution completely overlap at 0.237, confirming that the gap with the analytical solution is identical for both methods and is physical rather than numerical.

Norm conservation Both numerical solvers conserve the norm to machine precision: $\delta N_{\text{eigen}} = 1.8 \times 10^{-15}$ and $\delta N_{\text{CN}} = 2.9 \times 10^{-15}$. The analytical norm drifts by 2.7×10^{-2} over the simulation window, showing probability leaking past the walls into free space.

Computational cost The eigenvalue method requires 1.17 s to diagonalize the full 1999×1999 matrix upfront. Crank-Nicolson completes in 0.018 s, one LU factorization followed by 200 cheap triangular solves, making it $65\times$ faster while achieving essentially identical accuracy.

Ground state energy The finite-difference Hamiltonian gives $\tilde{E}_0 = 0.5004$, compared to the exact value $E_0 = 0.5000$, a relative error of 8×10^{-4} , consistent with the $\mathcal{O}(\Delta x^2)$ discretization error.

Table 4.3: Test 1 summary metrics.

Method	Final L^2 vs eigen	Norm drift	Runtime (s)
Eigenvalue (ref.)	—	1.8×10^{-15}	1.17
Crank-Nicolson	9.2×10^{-6}	2.9×10^{-15}	0.018
Analytical	0.237	2.7×10^{-2}	—

4.5 Test 2: Baseline PINN

4.5.1 PINN Architecture and Training

The Physics-Informed Neural Network represents the complex wave function as $\psi_\theta(x, t) = u_\theta(x, t) + iv_\theta(x, t)$, where both real and imaginary parts are outputs of a single fully connected network. The network is trained by minimizing:

$$\mathcal{L} = \mathcal{L}_{\text{PDE}} + w_{\text{IC}} \mathcal{L}_{\text{IC}} + w_{\text{BC}} \mathcal{L}_{\text{BC}}, \quad (4.11)$$

with $w_{\text{IC}} = w_{\text{BC}} = 100$. If the network starts from the wrong initial state, nothing that follows can be trusted. All derivatives are computed by automatic differentiation, PyTorch computes $\partial_t u$, $\partial_{xx} u$, $\partial_t v$, $\partial_{xx} v$ exactly by differentiating through the network.

Training proceeds in two phases. Adam runs for 2000 iterations, it knows which direction is downhill but not how steep the slope is, so it takes small cautious steps and can oscillate near the minimum. L-BFGS then takes over for 300 iterations, it approximates the curvature of the loss landscape and takes larger, smarter steps, making it much better at fine-tuning near the minimum. The two-phase strategy combines global exploration with local precision [11].

Table 4.4 summarizes the baseline configuration. This version uses no hard con-

straints, no Fourier features, and no norm penalty, the simplest possible formulation.

Table 4.4: Baseline PINN configuration (Test 2).

Parameter	Value
Hidden layers	4
Hidden units	64
Activation	tanh
Hard constraints	No (soft only)
Fourier features	No
Norm penalty	No
Collocation points	8000
IC/BC points	512 each
Adam iterations	2000
Adam learning rate	10^{-3}
L-BFGS iterations	300

4.5.2 Results

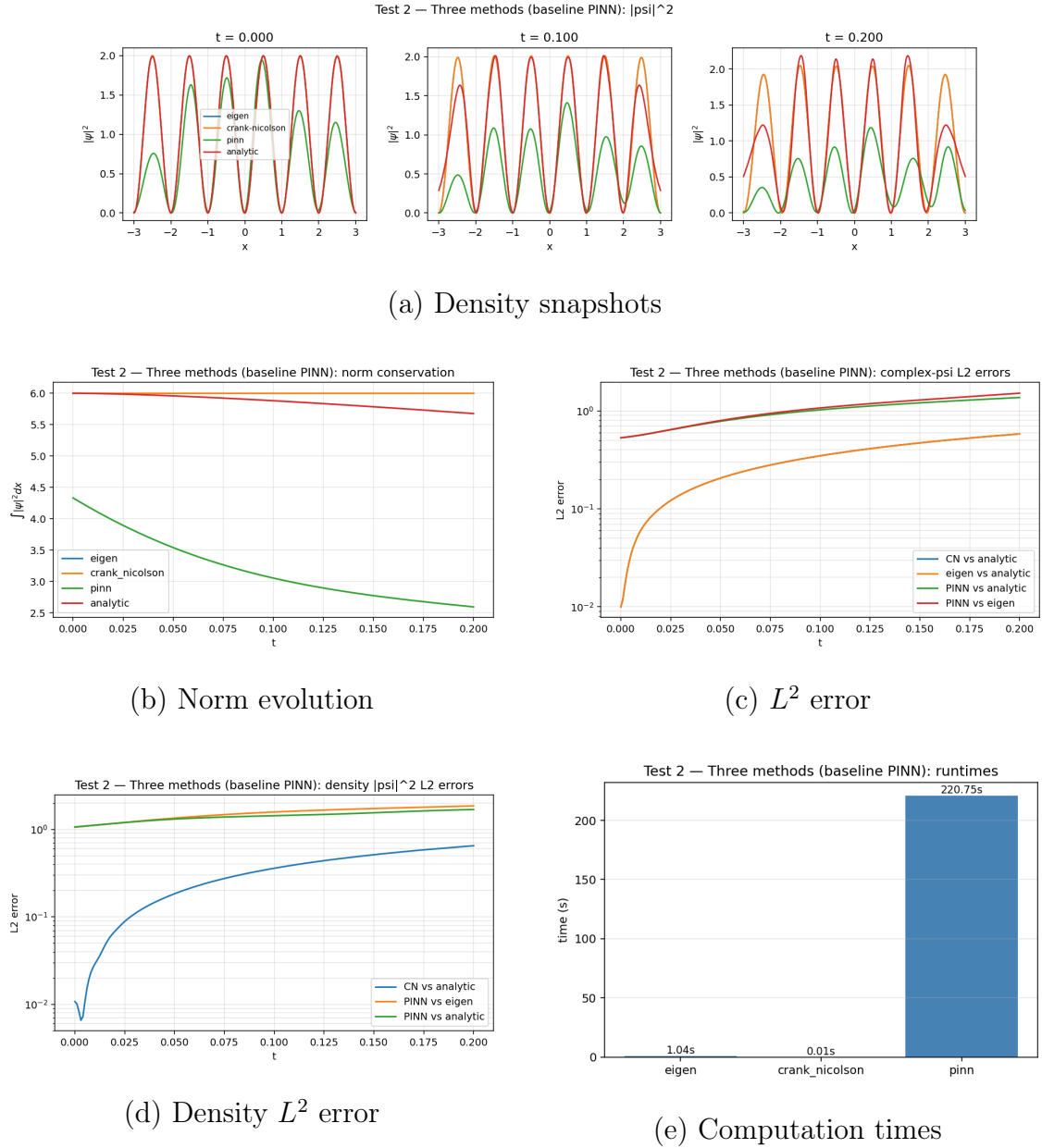


Figure 4.6: Test 2 — Baseline PINN vs eigenvalue method and Crank-Nicolson.

Wave function evolution At $t = 0$, eigen and CN overlap perfectly with peaks at the correct amplitude. The baseline PINN already starts wrong, its density peaks sit at roughly half the correct amplitude, meaning the soft IC constraint failed immediately. By $t = 0.1$ and $t = 0.2$ the PINN solution shows wrong amplitude and wrong shape throughout.

Norm conservation Eigen and CN conserve the norm to 1.8×10^{-15} and 3.8×10^{-14} respectively. The baseline PINN starts at a norm of ≈ 4.3 instead of 6, and collapses to ≈ 2.6 by $t = 0.2$, losing 40% of total probability. The network never satisfied the initial condition and cannot conserve probability at any point.

Accuracy The final L^2 error against the eigenvalue reference is 1.515, of order 1, no better than a random guess. The density L^2 reaches 1.853. Even after removing the global phase, the error remains 1.234, the failure is not a phase shift, it is a complete collapse of the wave function shape. The final training loss is 13.20, the optimizer never converged.

Table 4.5: Test 2 summary metrics.

Method	L^2 vs eigen	Density L^2	Norm drift	Time (s)
Eigenvalue (ref.)	—	—	1.8×10^{-15}	1.04
Crank-Nicolson	9.2×10^{-6}	—	3.8×10^{-14}	0.011
Baseline PINN	1.515	1.853	1.736	220.75

4.5.3 Why the Baseline PINN Fails

The failure is structural, not accidental. Several compounding reasons explain it.

Soft constraints cannot enforce the initial condition Without hard constraints, the IC is enforced only through a penalty term. The optimizer trades off IC accuracy against PDE accuracy and never truly satisfies $\psi(x, 0) = \sqrt{2} \sin(\pi x)$. Starting from the wrong initial state, the entire evolution is compromised.

Spectral bias A standard tanh network with 4 layers and 64 units is biased towards smooth, low-frequency functions. The initial condition is a rapidly oscillating

function that a vanilla MLP cannot represent accurately, not because the network is too small, but because its inductive bias works against high-frequency content. Without Fourier features, the network cannot learn the correct oscillatory structure.

No norm penalty Nothing in the loss explicitly prevents probability from leaking. The PDE residual alone is insufficient, the network can satisfy the Schrödinger equation approximately while allowing the norm to drift catastrophically.

A deeper theoretical reason Beyond these practical issues, the baseline PINN is attempting to solve an ill-posed problem. As shown by Shumaylov et al. [52], for non-chaotic systems such as the harmonic oscillator, trajectory data alone is provably insufficient to uniquely identify the governing dynamics. Soft penalty terms are not genuine physical priors, they can be satisfied superficially without capturing the true physics. Hard physical constraints are needed to make the problem well-posed, which is what Test 3 addresses.

4.6 Test 3: Improved PINN

4.6.1 Architectural Improvements

The improved PINN introduces four key changes over the baseline, each targeting a specific failure mode identified in Test 2.

Fourier features Before the network sees x and t , they are transformed into a higher-dimensional representation using sines and cosines at multiple frequencies. Neural networks are biased towards smooth, low-frequency functions by default and struggle to learn oscillatory solutions [46]. Fourier features provide pre-built oscillatory representations that the network can work with directly. This is one of the key improvements.

Hard constraints Instead of penalizing violations of the initial and boundary conditions, they are enforced exactly by construction. The network output $\mathcal{N}(x, t)$ is wrapped in the formula:

$$\psi_\theta(x, t) = \psi_0(x) + t(x - x_{\min})(x - x_{\max})\mathcal{N}(x, t), \quad (4.12)$$

which guarantees $\psi_\theta(x, 0) = \psi_0(x)$ and $\psi_\theta(\pm 3, t) = 0$ exactly, regardless of what the network learns. The baseline PINN was free to output anything at $t = 0$ and the boundaries, the optimizer merely tried to push them toward the correct values. Here they are forced directly into the output formula: the network cannot violate them even if it tries.

Norm penalty A penalty term is added to the loss to explicitly prevent probability from leaking:

$$\mathcal{L}_{\text{norm}} = \frac{1}{N_t} \sum_k \left(\int_{-3}^3 |\psi_\theta(x, t_k)|^2 dx - 1 \right)^2. \quad (4.13)$$

Additional training improvements The learning rate follows cosine annealing, decreasing gradually to aid convergence. Gradient clipping prevents exploding gradients early in training. RAD resampling moves collocation points every 1000 steps toward regions where the residual is highest, focusing training effort where the network is most wrong [53].

Table 4.6 summarizes the full configuration. The network is also larger (5 layers, 96 units vs 4 layers, 64 units) and trained longer (4000 Adam + 600 L-BFGS vs 2000 + 300).

Table 4.6: Improved PINN configuration (Test 3).

Parameter	Value
Hidden layers	5
Hidden units	96
Activation	tanh
Hard constraints	Yes
Fourier features	Yes (24 features, scale 2.0)
Norm penalty	Yes (weight 1.0)
Collocation points	15000
IC/BC points	512 each
Adam iterations	4000
Adam learning rate	10^{-3}
L-BFGS iterations	600

4.6.2 Results

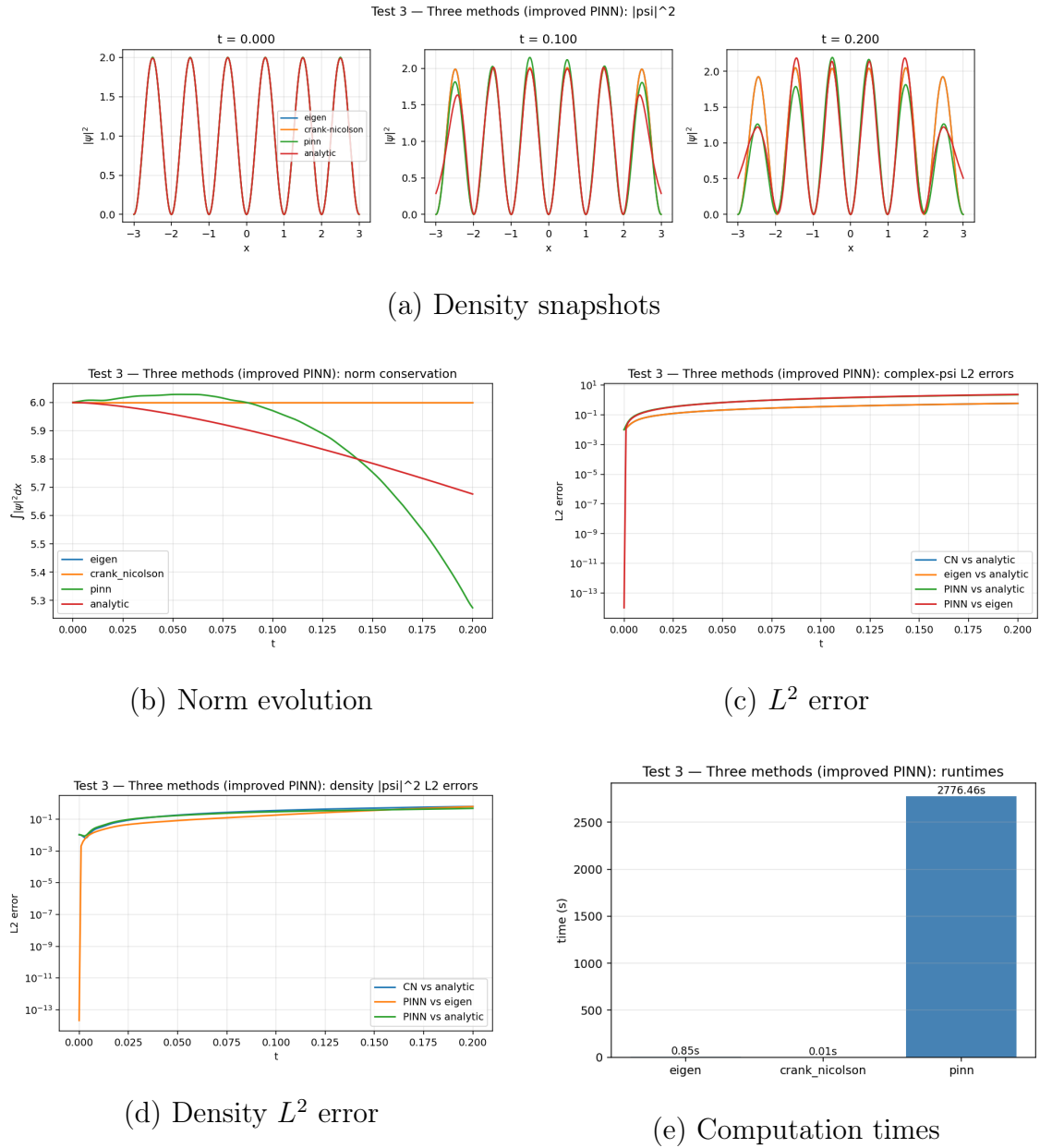


Figure 4.7: Test 3 — Improved PINN vs eigenvalue method and Crank-Nicolson.

Wave function evolution At $t = 0$, all four methods now overlap perfectly, the hard constraints are working exactly. At $t = 0.1$ and $t = 0.2$, the improved PINN is visually close to eigen and CN, with correct shape and approximately correct amplitude. Some deviation is visible near the boundaries and the peak at $t = 0.2$, but the overall evolution is qualitatively correct, a dramatic improvement over Test

2.

Accuracy The complex L^2 error of the PINN against eigen is 2.324, seemingly worse than the baseline. However, this number is misleading: it is dominated by a global phase offset, which is physically unobservable. A global phase $e^{i\theta}$ does not affect the probability density, energy, or any measurable quantity. The density L^2 error, the physically meaningful metric is 0.632, comparable to the CN vs analytic gap of 0.649. The phase-aligned L^2 , which removes the best global phase rotation before computing the error, gives 0.333. The improved PINN is as accurate as CN when measured against physical observables.

Norm conservation Eigen and CN remain flat at their respective machine-precision levels. The improved PINN starts correctly at the right norm and remains stable until $t \approx 0.1$, then drifts, reaching a norm drift of 0.726 by $t = 0.2$. This is much better than the baseline (1.736), but the norm penalty has not fully solved the conservation problem.

Training loss The final training loss is 0.400, three orders of magnitude below the baseline value of 13.20. The optimizer converged properly this time.

Table 4.7: Test 3 summary metrics.

Method	Density L^2 vs eigen	Phase-aligned L^2	Norm drift	Runtime (s)
Eigenvalue (ref.)	—	—	1.8×10^{-15}	0.85
Crank-Nicolson	0.649	—	3.8×10^{-14}	0.011
Improved PINN	0.632	0.333	0.726	2776.5

4.6.3 What Changed and What Didn't

The improvements are targeted. Hard constraints fixed the initial condition failure completely, the PINN now starts correctly at $t = 0$ by construction. Fourier features allowed the network to represent the oscillatory wave function accurately. The norm penalty reduced the probability leakage significantly, though not eliminated it.

What remains is the norm drift at later times and the computational cost. The improved PINN takes 2776 seconds which is approximately 46 minutes compared to 0.011 seconds for CN, to produce a solution that is comparable in density accuracy but still suffers from norm drift. The improvements moved the PINN from complete failure to physically meaningful results, but the gap in efficiency with classical methods remains enormous.

It is worth noting that all improvements were applied simultaneously, so it is not possible from this experiment alone to isolate the contribution of each change. An ablation study removing one improvement at a time would be needed to quantify individual contributions, this is left as a limitation of the current benchmark.

4.7 Supplementary Experiment: High-Capacity PINN at $T=2.0$

This experiment is not part of the three main comparisons but is worth reporting as a supplementary result. It tests the most powerful PINN configuration on a ten times longer time window, on GPU, to probe the limits of the approach.

Table 4.8: Architecture comparison: Test 3 improved PINN vs standalone GPU PINN.

Parameter	Test 3 (CPU)	Standalone (GPU)
Hidden layers	5	6
Hidden units	96	128
Activation	tanh	swish
Fourier mapping size	24	128
Fourier scale	2.0	5.0
Norm penalty weight	1.0	2.0
Collocation points	15000	16384
Adam iterations	4000	15000
L-BFGS iterations	600	500
RAD resampling	Yes	Yes
Early stopping	No	Yes
Time window T	0.2	2.0
Hardware	CPU	GPU (RTX A4000)

4.7.1 Training Convergence

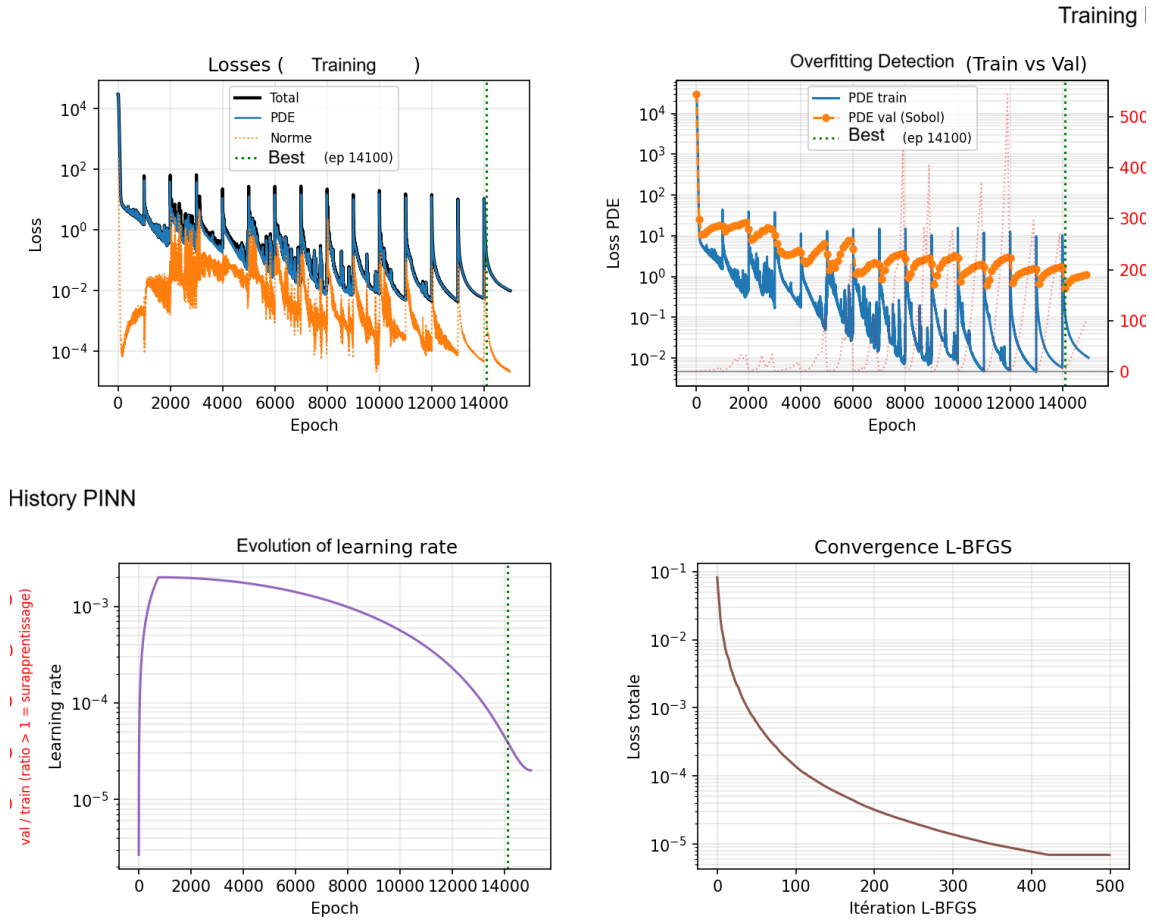


Figure 4.8: Training history of the standalone GPU PINN: training losses and train-validation behaviour on the first row; learning-rate schedule and L-BFGS convergence on the second row.

The loss started at $\sim 30,000$ at epoch 0 and dropped to 0.010 by epoch 15000. The regular spikes every 1000 epochs are the RAD resampling which is healthy and expected. IC and BC losses are exactly zero throughout, confirming hard constraints are working.

The train vs validation gap reveals partial overfitting: train loss reaches $\sim 10^{-2}$ while validation stays higher. Early stopping identified the best model at epoch 14100 but was never triggered, the network kept improving until the very end and could have benefited from more iterations.

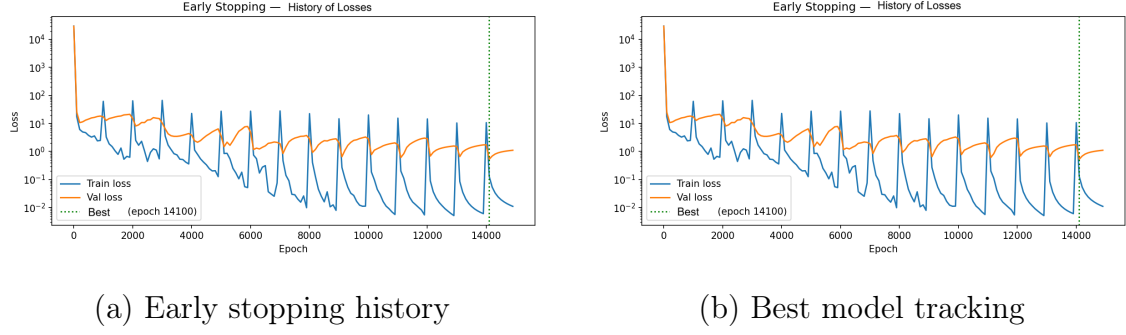


Figure 4.9: Train vs validation loss with best model marker at epoch 14100.

L-BFGS converged from 0.082 at iteration 0 to 6.97×10^{-6} by iteration ≈ 450 , then plateaued, four orders of magnitude reduction. Adam found a good basin and L-BFGS found the precise bottom of it.

4.7.2 Results at $T=2.0$

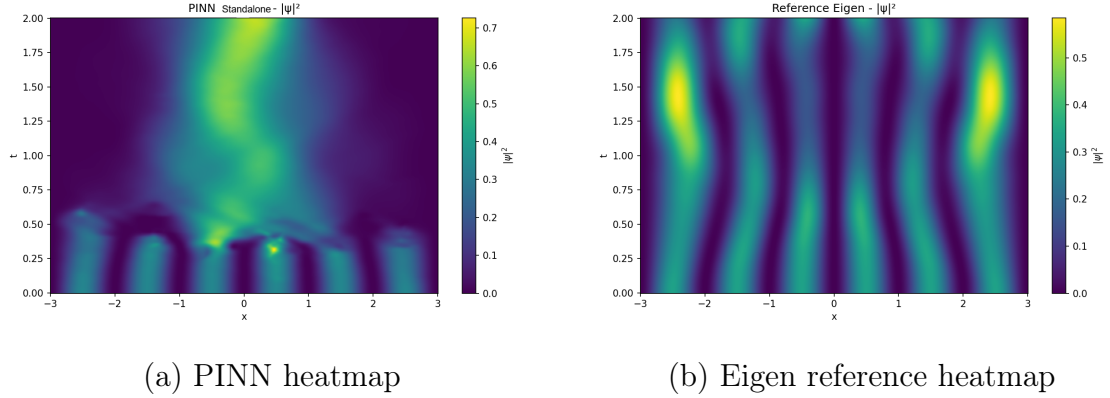


Figure 4.10: Probability density $|\psi(x, t)|^2$ over $T = 2.0$.

The eigen heatmap is clean and symmetric, coherent quantum oscillations with structured banding. The PINN heatmap is chaotic and asymmetric, with probability concentrating near $x = 0$ at later times with no physical basis. The PINN lost track of the correct evolution entirely after $t \approx 0.5$.

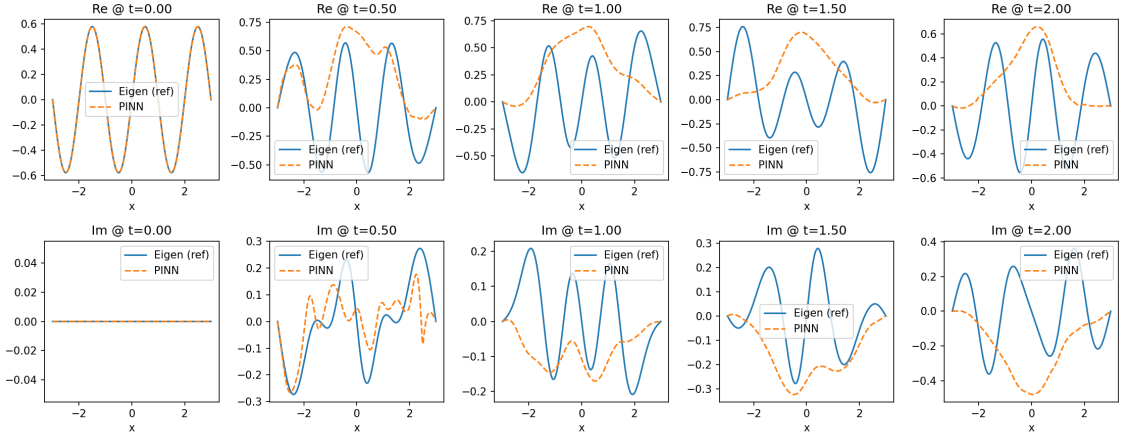
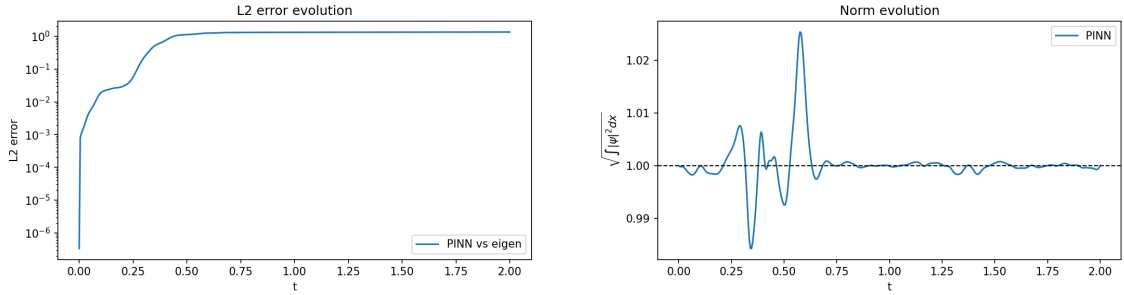


Figure 4.11: Real and imaginary parts of $\psi(x, t)$ at selected time instants: eigen (blue) vs PINN (orange).

At $t = 0$ both overlap perfectly. From $t = 0.5$ onward divergence grows progressively, the real part has the right number of oscillations but wrong amplitude and phase, while the imaginary part is entirely wrong by $t = 1.0$.



(a) L2 error over time

(b) Norm conservation

Figure 4.12: Accuracy and norm conservation of the standalone GPU PINN over $T = 2.0$.

The L2 error starts near machine precision at $t = 0$, grows rapidly to $\sim 10^0$ by $t = 0.5$, then saturates there for the rest of the window, the PINN is stuck at this accuracy level and never recovers. The norm oscillates around 1.0 with $\approx 2.5\%$ amplitude, then settles to near-perfect conservation after $t = 1.0$, giving a final norm drift of 0.025, the best across all PINN experiments.

Table 4.9: Standalone GPU PINN metrics at $T = 2.0$.

Metric	Value
Runtime	2479 s
Mean L2 error	1.077
Max L2 error	1.358
Mean density L2	0.503
Norm drift	0.025
L-BFGS final loss	6.97×10^{-6}
Best epoch (Adam)	14100

4.7.3 Why the PINN Fails at Longer Time Horizons

This is not a training failure, the optimizer did everything right.

The PINN learns the entire space-time domain at once. For $T = 2.0$, the wave function has gone through many rapid oscillation cycles, developing complex spatial structure that grows in complexity with time. The architecture got bigger, but the problem became ten times harder, and the extra capacity was not enough to compensate.

CN does not have this problem. It takes more steps of the same cheap operation, difficulty does not grow with time. The PINN must represent the entire trajectory simultaneously, and the cost of that representation grows with T .

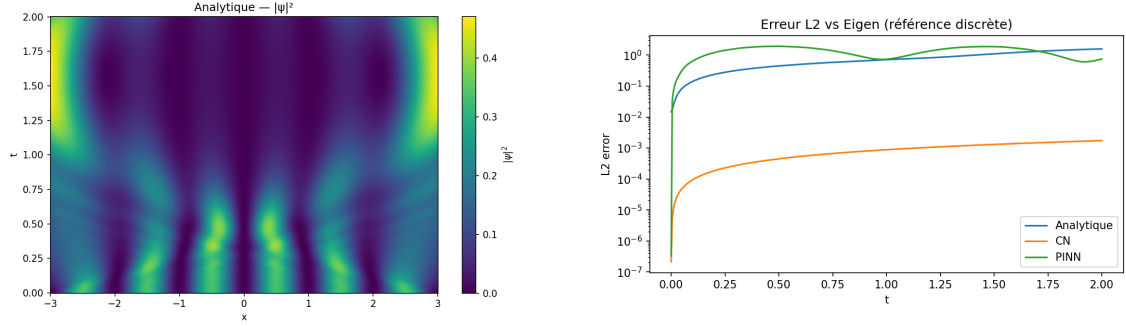
Table 4.10: Norm drift across all PINN experiments.

Experiment	Norm drift	Time window
Baseline PINN (Test 2)	1.736	$T = 0.2$
Improved PINN (Test 3)	0.726	$T = 0.2$
Standalone GPU PINN	0.025	$T = 2.0$

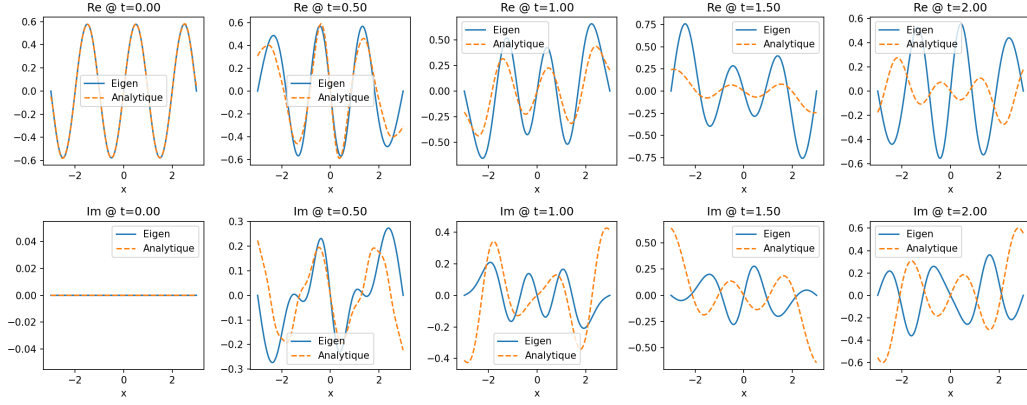
The norm penalty weight of 2.0 combined with 15000 training epochs solved the norm conservation problem, two orders of magnitude better than the baseline. But norm conservation and wave function accuracy are partially orthogonal objectives: even with near-perfect norm conservation, the mean L2 error remains 1.077.

4.8 Supplementary Experiment: All Methods at $T=2.0$

This experiment runs all methods fresh at $T = 2.0$ to confirm whether the conclusions from Tests 1–3 hold at longer time horizons. It is supplementary and optional.

(a) Analytical at $T = 2.0$

(b) L2 error vs eigen



(c) Analytical vs eigen Re/Im

Figure 4.13: Comparative results at $T = 2.0$ for all methods.

Analytical solution (a) Instead of clean oscillating bands, the free-space solution disperses across the domain, concentrating near the walls at $x = \pm 3$ by $t \approx 1.75$. This is the wave packet leaking past the walls, physically correct for the infinite harmonic oscillator, wrong for the box problem. The box-versus-free-space effect becomes dramatic at $T = 2.0$.

L2 error over time (b) CN stays flat at $\sim 10^{-3}$ throughout, no degradation over the longer window. The analytical grows steadily to $\sim 10^0$ by $T = 2.0$, physical divergence, not a solver error. The PINN spikes to $\sim 10^0$ by $t = 0.25$, oscillates, and partially recovers to ≈ 0.75 by $T = 2.0$, interesting but still far from accurate.

Analytical vs eigen Re/Im (c) At $t = 0$ both overlap. From $t = 0.5$ onward progressive divergence appears in both real and imaginary parts. By $t = 1.0$ the analytical imaginary part has a larger amplitude than eigen, and by $t = 2.0$ both parts differ completely in phase, amplitude, and number of oscillations, the most dramatic visualization of the physical box-versus-free-space difference.

Table 4.11: All methods at $T = 2.0$.

Method	Mean L2	Norm drift	Runtime (s)
Eigen (ref.)	—	2.5×10^{-14}	0.68
CN	8.8×10^{-4}	9.3×10^{-15}	0.016
Analytical	0.769	0.347	0.042
PINN	1.302	0.025	1955

At $T = 2.0$, CN remains three orders of magnitude more accurate than the PINN (8.8×10^{-4} vs 1.302) while being 122,000 $\times$ faster. The performance gap between CN and PINN widens at longer time horizons, confirming that the PINN’s difficulties are fundamental and not resolved by longer training or a more powerful architecture.

4.9 Summary: Forward Problem

The forward experiments lead to four clear conclusions.

Classical methods are reliable and efficient for simple systems The eigenvalue method and Crank-Nicolson both solve the box problem faithfully, conserving

the norm to machine precision and agreeing with each other to 9.2×10^{-6} in L^2 norm. CN achieves this at a fraction of the cost, 0.016 s vs 0.85 s, making it the method of choice for 1D forward simulation on regular grids.

The baseline PINN is a complete failure Without hard constraints, Fourier features, and norm penalty, the PINN never satisfies the initial condition, loses a large chunk of total probability, and produces a solution of order 1 in L^2 error. The failure is structural, rooted in spectral bias, soft constraints, and an ill-posed formulation.

Targeted improvements recover physical accuracy Hard constraints, Fourier features, and a norm penalty transform the PINN from complete failure to physically meaningful results. The improved PINN matches CN in density L^2 accuracy (0.632 vs 0.649), but at a much higher computational cost: the PINN requires 2776 s of training against 0.011 s for the full CN solve, i.e. a ratio of $2776/0.011 \approx 2.5 \times 10^5$, so the PINN is roughly 250,000 times more expensive for comparable accuracy on this problem.

PINN accuracy degrades at longer time horizons At $T = 2.0$, the PINN mean L^2 error reaches 1.302 while CN stays at 8.8×10^{-4} , an error ratio of $1.302/(8.8 \times 10^{-4}) \approx 1.5 \times 10^3$, i.e. the PINN error is roughly 1500 times larger. This is a fundamental limitation: the PINN must represent the entire space-time trajectory simultaneously, and the complexity of that representation grows with T . CN is indifferent to time horizon.

The PINN's advantage is not forward simulation accuracy on 1D problems, classical methods dominate here. Its potential lies in problems where classical methods fail like higher dimensions, irregular geometries, and the inverse problem, which is the subject of the next part.

Chapter 5

Implementation of Inverse Solvers and Results

5.1 Problem Setup and Data Generation

5.1.1 Problem Setup

The inverse problem consists of recovering the unknown potential $V(x)$ from noisy observations of the wave function $\psi(x, t)$, without any prior knowledge of its form. The true potential $V_{\text{true}}(x) = \frac{1}{2}x^2$ is what both methods attempt to reconstruct. Two methods are benchmarked: Tikhonov regularization and the inverse PINN, which trains two networks jointly, one for $\psi(x, t)$ and one for $V(x)$, coupled through the PDE residual.

The physical system is the same harmonic oscillator, but several parameters differ from the forward problem, as summarized in Table 5.1.

Table 5.1: Parameter comparison: forward vs inverse problem.

Parameter	Forward	Inverse
Spatial domain	$[-3, 3]$	$[-6, 6]$
Spatial points N_x	2001	257
Time domain	$[0, 0.2]$	$[0, 2\pi]$
Time steps N_t	201	401
Initial condition	$\sqrt{2} \sin(\pi x)$	$(\frac{1}{\pi})^{1/4} e^{-(x-1)^2/2}$
Noise level	None	1% complex Gaussian
Objective	Solve for ψ	Recover $V(x)$

The wider domain $[-6, 6]$ is necessary because the Gaussian wave packet oscillates coherently and spreads further than the sinusoidal state, a narrower domain would cause boundary artifacts. The time window $T = 2\pi/\omega$ corresponds to exactly one classical oscillation period: the wave packet returns to its starting position, meaning the inverse methods observe the complete oscillatory behavior of the system and have maximum information about the potential shape.

The Gaussian initial condition is chosen deliberately, it excites many eigenmodes simultaneously, illuminating the potential across a wider spatial range and giving both methods more information to work with than a single-mode sinusoidal state would.

All reconstruction errors are evaluated on the illuminated zone $\Omega_{\text{eval}} = \{x : |x| \leq 3\}$, where $|\psi|^2$ is significant. Outside this zone, $|\psi| \rightarrow 0$ and the data provide no physical constraint on V , any reconstruction there reflects the regularization prior,

not the observations. The metric is the relative L^2 error:

$$\mathcal{E}_{\text{rel}} = \frac{\|V_{\text{rec}} - V_{\text{true}}\|_{L^2(\Omega_{\text{eval}})}}{\|V_{\text{true}}\|_{L^2(\Omega_{\text{eval}})}}. \quad (5.1)$$

5.1.2 Data Generation

The reference wave function is generated by running Crank-Nicolson with the true harmonic potential, producing a clean solution $\psi_{\text{clean}}(x, t)$. Complex Gaussian noise is then added:

$$\psi_{\text{obs}}(x, t) = \psi_{\text{clean}}(x, t) + \eta \max_{x, t} |\psi_{\text{clean}}| (\xi_r + i \xi_i), \quad \xi_r, \xi_i \sim \mathcal{N}(0, 1), \quad (5.2)$$

with $\eta = 1\%$. This is what makes the inverse problem ill-posed: both methods must recover $V(x)$ from ψ_{obs} alone, without ever seeing V_{true} or ψ_{clean} .

The output is a 401×257 complex matrix, 401 time steps and 257 spatial points. This CN solution is the ground truth against which both inverse methods are benchmarked. The quality of this forward simulation determines the ceiling of what the inverse methods can achieve: clean and accurate forward data enables good reconstruction; high noise degrades it. Norm conservation is verified throughout to confirm the forward solver worked correctly.

5.2 Tikhonov Regularization

5.2.1 Implementation

Tikhonov regularization recovers $V(x)$ by rearranging the Schrödinger equation directly. At every space-time point (x_j, t_k) , the equation gives:

$$V(x_j) \cdot \psi(x_j, t_k) = i\hbar \partial_t \psi + \frac{\hbar^2}{2m} \partial_{xx} \psi = R(x_j, t_k). \quad (5.3)$$

Stacking all these equations yields the linear system $AV = b$, where each row of A is $\psi(x_j, t_k)$ at one point and each entry of b is the right-hand side derivative term. Rows where $|\psi| < 2\%$ of the maximum are discarded, in regions where $\psi \approx 0$, the equation carries no information about V .

Before assembling the system, the noisy observations are smoothed using a Savitzky-Golay filter [54], a polynomial fitted locally to remove noise while preserving the signal shape. Without this step, numerical differentiation of noisy data amplifies noise catastrophically, making the right-hand side b unreliable.

The system is solved by minimizing:

$$\hat{V} = \arg \min_V \|AV - b\|_2^2 + \alpha \|LV\|_2^2, \quad (5.4)$$

where L is the discrete second derivative operator. Minimizing $\|LV\|_2^2$ penalizes potentials with large curvature, enforcing smoothness consistent with the known parabolic shape of V_{true} .

Regularization Parameter Selection

The regularization parameter α controls the trade-off between data fidelity and smoothness. Too small: the solution fits the noisy data and oscillates wildly. Too large: the solution is overly smooth and misses the true shape. We test 40 values logarithmically spaced from 10^{-10} to 10^2 and select the optimal α via Generalized Cross-Validation (GCV). GCV is preferred over Morozov's discrepancy principle because it does not require prior knowledge of the exact noise level, it estimates the optimal α purely from the data structure.

Variants

Four variants are benchmarked, each adding targeted improvements:

Table 5.2: Tikhonov variants.

Version	Differentiation	Regularization	α selection
v1	FD order 2	H^2	GCV 1D
v2	FD order 4	H^2	GCV 1D
v3	FD order 4 + 2D smoothing	$H^1 + H^2$	GCV 2D
v4	same as v3	$H^1 + H^2$	K-fold CV 2D

v2 uses a 5-point stencil for derivatives, reducing truncation error. v3 adds joint 2D Gaussian smoothing and a combined Sobolev regularization penalizing both slope (H^1) and curvature (H^2). v4 replaces GCV with K-fold cross-validation for more robust parameter selection. Results are reported in the comparisons with inverse PINNs later on.

5.3 Inverse PINN

5.3.1 Architecture

Unlike the forward PINN where $V(x)$ is known, the inverse problem requires learning both the wave function and the potential simultaneously. Two separate networks are used:

- $\psi_{\text{net}}(x, t)$ takes (x, t) as input, outputs a complex number $u + iv$. 4 layers, 128 units.
- $V_{\text{net}}(x)$ takes x only, outputs a real number. 4 layers, 64 units. Time-independent.

The two networks never touch each other architecturally. They meet only at the

PDE loss, where both outputs are used to compute the Schrödinger residual:

$$\mathcal{L}_{\text{PDE}} = \frac{1}{N_c} \sum_k \left| i\hbar \partial_t \psi_{\text{net}} + \frac{\hbar^2}{2m} \partial_{xx} \psi_{\text{net}} - V_{\text{net}} \cdot \psi_{\text{net}} \right|^2. \quad (5.5)$$

When the loss is backpropagated, gradients flow back into both networks simultaneously, they are trained jointly by one shared optimizer.

5.3.2 Loss Function and Physical Priors

The total loss is:

$$\mathcal{L} = w_{\text{data}} \mathcal{L}_{\text{data}} + w_{\text{PDE}} \mathcal{L}_{\text{PDE}} + w_{\text{IC}} \mathcal{L}_{\text{IC}}, \quad (5.6)$$

with $w_{\text{data}} = w_{\text{PDE}} = 1$ and $w_{\text{IC}} = 10$. Boundary conditions are not explicitly penalized, the Gaussian initial condition decays naturally near $x = \pm 3$, and the data loss implicitly enforces $\psi_{\text{obs}} \approx 0$ at the boundaries.

Two physical priors are built directly into V_{net} :

- **Symmetry:** $V_{\text{net}}(x) = V_{\text{net}}(-x)$, enforced by feeding $|x|$ to the network instead of x .
- **Zero at origin:** $V_{\text{net}}(0) = 0$, enforced by subtracting the network output at $x = 0$.

These priors are the inverse PINN's equivalent of Tikhonov's smoothness penalty, both inject prior knowledge to restore uniqueness. As shown by Shumaylov et al. [52], for non-chaotic systems such as the harmonic oscillator, soft penalties alone are insufficient; hard physical priors are mathematically necessary to make the inverse problem well-posed.

5.3.3 Training Configuration

Table 5.3: Inverse PINN v1 configuration.

Parameter	Value
ψ_{net} layers/units	4 / 128
V_{net} layers/units	4 / 64
Data points	5000
Collocation points	4000
IC points	200
Adam iterations	8000
Learning rate	2×10^{-3}
Symmetry prior	Yes
Zero at origin	Yes

Results for all variants are reported in the comparison.

5.4 Initial Comparison: Tikhonov v1 vs Inverse PINN v1

5.4.1 Reference Data

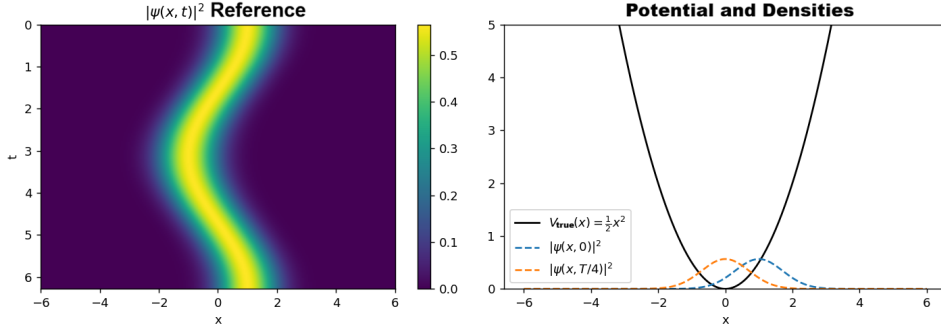


Figure 5.1: Forward reference data for the inverse problem. Left: probability density $|\psi(x, t)|^2$, the Gaussian wave packet oscillates coherently between $x = 1$ and $x = -1$ over one full period $T = 2\pi$. Right: true potential $V_{\text{true}}(x) = \frac{1}{2}x^2$ with density snapshots at $t = 0$ and $t = T/4$. Both snapshots remain well within $|x| \leq 3$, confirming that the illuminated zone captures all the physics.

5.4.2 Tikhonov v1

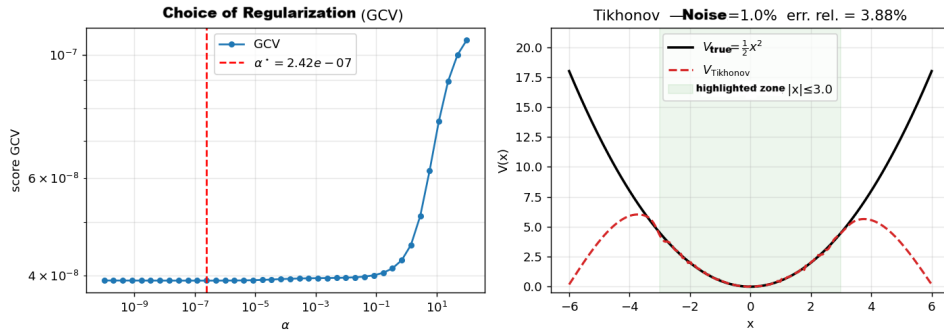


Figure 5.2: Tikhonov v1 (FD-2, GCV). Left: GCV score as a function of α — flat plateau up to $\alpha^* = 2.42 \times 10^{-7}$, then sharp rise. Right: reconstructed potential at 1% noise. Inside $|x| \leq 3$ the red dashed curve tracks the parabola closely; outside the illuminated zone it diverges. Relative L^2 error: 3.88%.

Inside the illuminated zone the reconstruction is excellent. The flat GCV plateau indicates that a wide range of α values below 10^{-7} all produce comparably good fits; the sharp rise beyond α^* marks the onset of over-smoothing where the regularization

term begins to dominate and distort the parabolic shape. Outside $|x| \leq 3$, the reconstruction diverges with oscillations peaking near $V \approx 6$ around $x = \pm 3.5$. This is expected: where $|\psi|^2 \approx 0$, the system $AV = b$ receives no data constraint on V , and the smoothness prior fills in a physically incorrect extrapolation.

5.4.3 Inverse PINN v1

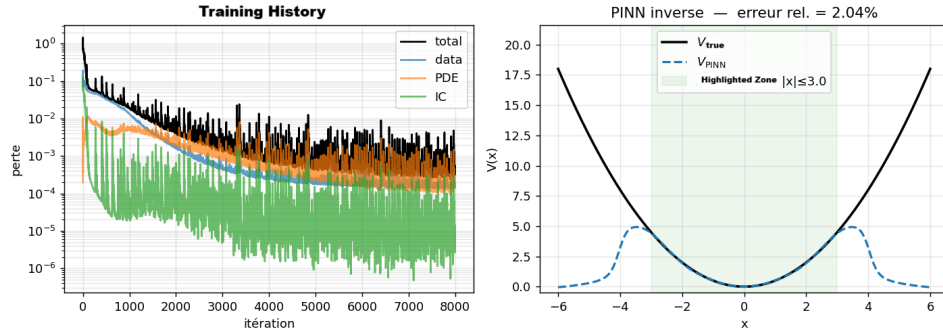


Figure 5.3: Inverse PINN v1 (dedicated run, 8000 Adam iterations). Left: training history, total loss (black) drops from ~ 1 to $\sim 10^{-3}$; IC loss (green) reaches the lowest floor due to its higher weight ($w_{\text{IC}} = 10$); data loss (blue) is the smoothest curve; PDE loss (orange) decreases with oscillations; spikes reflect random resampling of collocation points throughout training. Right: reconstructed potential at 1% noise. Inside $|x| \leq 3$ the blue dashed curve tracks the parabola closely; outside the illuminated zone it collapses to zero. Relative L^2 error: 2.04%.

The training history shows healthy convergence. The IC loss reaches the lowest floor because the initial condition is stabilized early by its higher weight and maintained throughout. The PDE loss decreases more slowly and with oscillations, reflecting the difficulty of satisfying the Schrödinger residual simultaneously across the full space-time domain. The spiky behavior is expected: random resampling of collocation points at each iteration prevents the network from overfitting to a fixed training set and keeps the loss landscape from stagnating. Outside $|x| \leq 3$, the PINN collapses to zero rather than oscillating. This is a direct consequence of the architectural priors, symmetry and zero-at-origin which anchor the network to a smooth function through the origin when no data is present, a qualitatively more physical extrapolation than Tikhonov’s oscillatory behavior, even if neither is

correct.

5.4.4 Direct Comparison

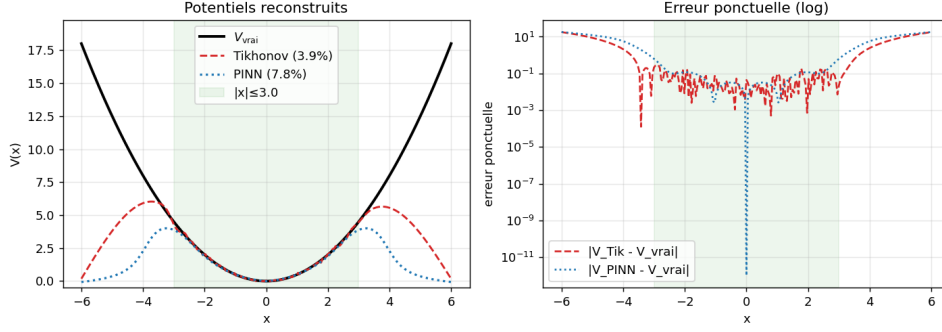


Figure 5.4: Tikhonov v1 (red dashed) vs inverse PINN v1 (blue dotted) against the true potential (black), using the automated benchmark configuration. Left: reconstructed potentials, both methods agree well inside $|x| \leq 3$ and diverge outside. Right: pointwise absolute error on log scale. Inside the illuminated zone, Tikhonov reaches lower pointwise errors at some locations (down to $\sim 10^{-4}$) but with strong oscillations; the PINN error is smoother, settling around 10^{-3} , with a sharp dip near $x = 0$ imposed by the zero-at-origin constraint. Both methods diverge sharply outside $|x| \leq 3$.

Table 5.4: Tikhonov v1 vs inverse PINN v1 at 1% noise, evaluated on $|x| \leq 3$.

Method	Rel. L^2 error	Runtime (s)
Tikhonov v1 (FD-2)	3.88%	0.60
Inverse PINN v1	2.04%	160

Both methods recover the potential accurately inside the illuminated zone, but the comparison requires careful interpretation.

Tikhonov’s limitation is numerical differentiation: even after Savitzky-Golay smoothing, differentiating noisy observations propagates noise into the right-hand side b , which the smoothness prior only partially suppresses. This explains the oscillatory pointwise error profile visible in Figure 5.4, where accuracy varies significantly across the illuminated zone despite the low global error.

The PINN avoids this, it learns a smooth function and differentiates through the network via automatic differentiation, never touching the noisy data directly. This accounts for its smoother error profile. However, its accuracy advantage is partly structural, the symmetry constraint and zero-at-origin condition are hard-wired into the architecture with prior knowledge of the true potential’s properties. For the harmonic oscillator these priors are exactly correct and significantly narrow the solution space. Without them as the benchmark run at 7.8% already shows, the PINN performs worse than Tikhonov. Tikhonov’s smoothness prior makes no such assumption and remains valid across a much wider class of problems.

In summary, Tikhonov v1 wins on speed, reproducibility, and generality; the PINN wins on accuracy (2.04% vs 3.88%) but at $267\times$ the computational cost and only because problem-specific structural knowledge was embedded in its architecture from the outset. The full benchmark examines whether this advantage holds as both methods are pushed further.

Figure 5.4 reports the PINN at 7.8%, not 2.04%. This corresponds to a lighter configuration used in the automated benchmark (fewer Adam iterations, smaller network). Both are PINN v1 in architecture; the discrepancy is purely a matter of training budget.

5.5 Benchmark: Tikhonov v1–v2 vs Inverse PINN v1–v2

5.5.1 Variant Configurations

Each method is improved from its v1 baseline along a specific axis. Table 5.5 summarizes the changes.

Table 5.5: Configuration changes from v1 to v2 for each method.

Parameter	v1	v2
<i>Tikhonov</i>		
FD stencil	Order 2 (3-point)	Order 4 (5-point)
Regularization	H^2 (curvature)	H^2 (curvature)
α selection	GCV 1D	GCV 1D
<i>Inverse PINN</i>		
Network size	48 units, tanh	128 units, tanh
Fourier features	No	Yes (40 features)
Optimizer	Adam only (8000)	Adam + L-BFGS
Norm penalty	No	Yes
Adaptive weights	No	Yes

5.5.2 Results

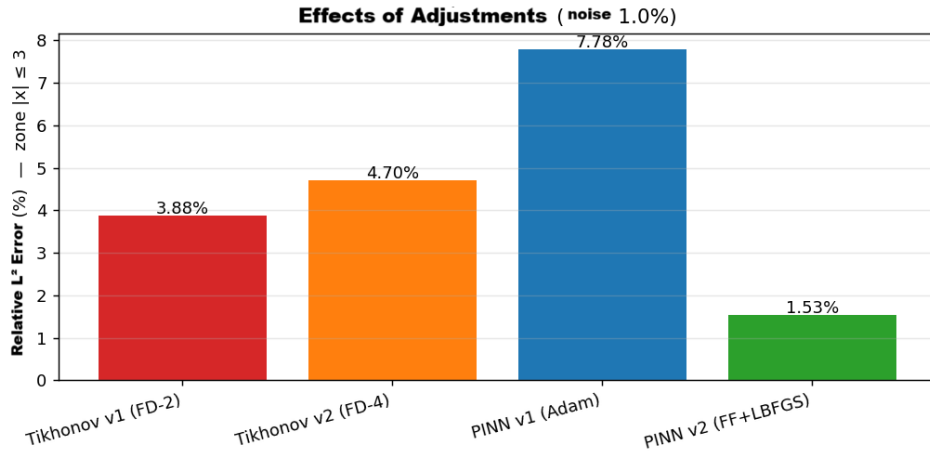


Figure 5.5: Relative L^2 error for all four variants at 1% noise, evaluated on $|x| \leq 3$. Lower is better. Tikhonov v2 regresses from v1 (4.70% vs 3.88%); PINN v2 improves dramatically over v1 (1.53% vs 7.78%) and achieves the best result overall.

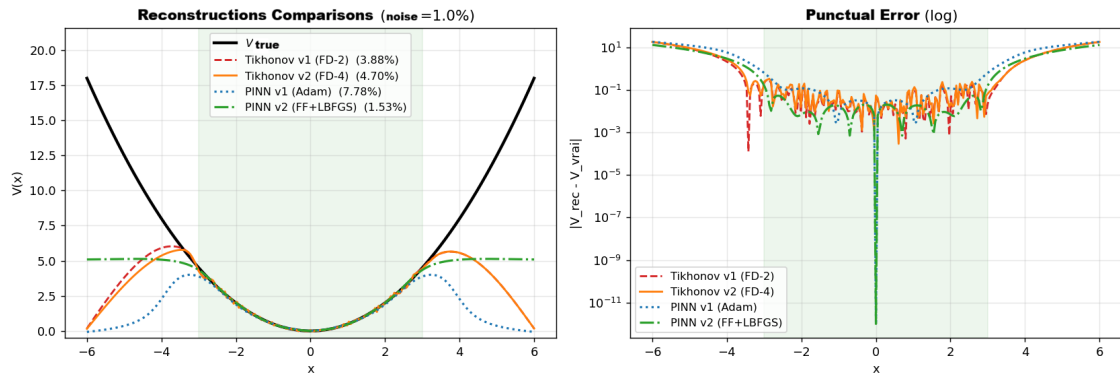


Figure 5.6: All four variants at 1% noise. Left: reconstructed potentials, all methods agree well inside $|x| \leq 3$ except PINN v1 (blue dotted), which undershoots the parabola; PINN v2 (green) plateaus at $V \approx 5$ outside the illuminated zone rather than oscillating or collapsing. Right: pointwise absolute error on log scale. Inside the zone, PINN v2 and Tikhonov v1 achieve the lowest and smoothest errors; Tikhonov v2 is noisier than v1 despite its higher-order stencil.

Table 5.6: Full benchmark results at 1% noise, evaluated on $|x| \leq 3$.

Method	Rel. L^2 error	Runtime (s)
Tikhonov v1 (FD-2)	3.88%	0.60
Tikhonov v2 (FD-4)	4.70%	0.59
PINN v1 (Adam)	7.78%	160
PINN v2 (FF+LBFGS)	1.53%	275

5.5.3 Analysis

The results reveal a counterintuitive pattern on both sides: added complexity does not always help, and the reasons differ between methods.

Tikhonov: higher-order derivatives hurt Switching from a 3-point to a 5-point stencil was expected to reduce truncation error in the derivatives. Instead, accuracy dropped from 3.88% to 4.70%. The explanation lies in the noise regime: at 1% noise, the dominant source of error is not truncation error but noise amplification during numerical differentiation. A higher-order stencil uses more neighboring points, which means it also incorporates more noise from a wider spatial window. The Savitzky-Golay smoothing and H^2 regularization were tuned around the simpler 3-point stencil; the 5-point version amplifies residual noise faster than it reduces truncation error, producing a noisier right-hand side b and a worse reconstruction. The pointwise error plot confirms that Tikhonov v2 (orange) is visibly more oscillatory than v1 (red) inside the illuminated zone.

PINN: targeted improvements deliver the best result PINN v2 improves from 7.78% to 1.53%, the largest gain across the benchmark. Three changes drove this. Fourier features removed the spectral bias that prevented v1 from represent-

ing the oscillatory wave function accurately, giving the network the right inductive bias from the start. L-BFGS refined the solution after Adam found a good basin, exploiting second-order curvature information to reach a sharper minimum. The norm penalty explicitly prevented probability from leaking during training, stabilizing the joint optimization of ψ_{net} and V_{net} . Outside $|x| \leq 3$, PINN v2 plateaus at $V \approx 5$ rather than collapsing to zero or oscillating, a side effect of the norm penalty anchoring the network output at a non-zero value in the data-free region.

Overall verdict PINN v2 achieves the best accuracy (1.53%) but at 275 s, roughly $460\times$ slower than Tikhonov v1. Among classical methods, Tikhonov v1 remains the best performer, the simpler FD stencil is better adapted to the noise level. The key finding is that improvement is not monotone, on both sides, the v2 variant that added complexity produced a counterintuitive outcome, regression for Tikhonov and a large gain for the PINN, showing how differently the two methods respond to the same noise level and the same inverse problem.

So the PINN is either the best or the worst depending on configuration. That’s actually the more interesting finding: the PINN has higher variance, it’s more sensitive to its setup than Tikhonov. Tikhonov v1 and v2 stay in the 3.88

5.6 Tikhonov Robustness to Noise

The benchmark results so far were obtained at a fixed noise level of 1%. To assess how sensitive each Tikhonov variant is to the noise level, both v1 and v2 are evaluated across six noise levels: 0%, 0.1%, 0.5%, 1%, 2%, and 5%. At zero noise both variants should recover the potential near-perfectly; the question is how fast the error grows and which variant degrades more gracefully as noise increases.

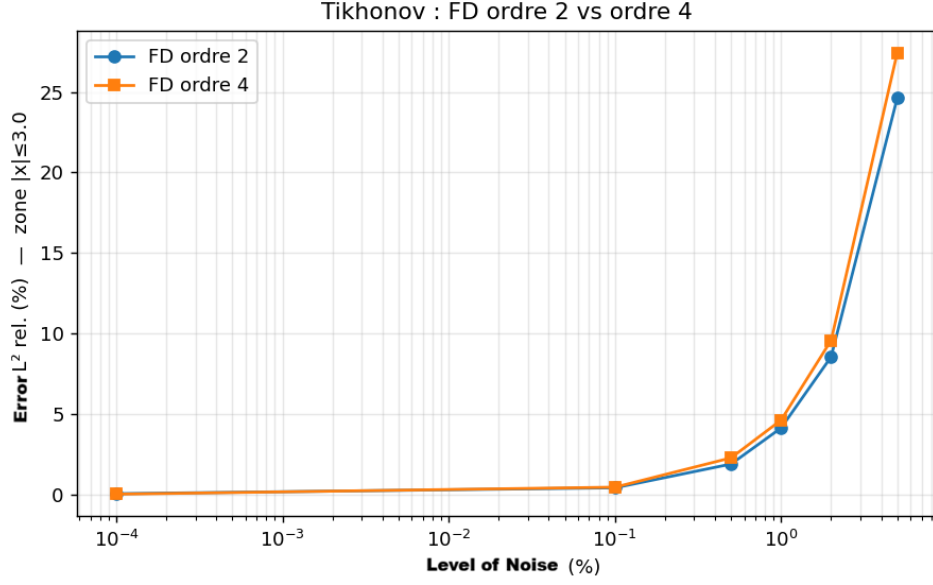


Figure 5.7: Relative L^2 error of Tikhonov v1 (FD-2) and v2 (FD-4) as a function of noise level, evaluated on $|x| \leq 3$.

Both variants recover the potential near-perfectly at low noise: error stays below 0.5% up to $\eta = 0.1\%$. Beyond that, both degrade at the same rate as noise increases, reaching $\sim 4\%$ at $\eta = 0.5\%$, $\sim 9\%$ at $\eta = 2\%$, and $\sim 25\%$ at $\eta = 5\%$.

The two curves are nearly indistinguishable throughout. FD-4 is marginally worse at every noise level above 0.1%, confirming the finding from the benchmark: in the noise-dominated regime, the higher-order stencil offers no advantage and introduces slightly more noise amplification. Both variants degrade gracefully with no catastrophic failure, confirming that GCV parameter selection remains effective across the full noise range tested.

5.7 Benchmark v3: Six-Method Comparison

5.7.1 New Adjustments

Tikhonov v3 and PINN v3 extend the previous versions with several improvements designed to make the reconstruction smoother, more stable, and more physically consistent. Rather than introducing entirely new methods, these additions

refine how the algorithms learn from noisy data and enforce the underlying physics.

Sobolev $H^1 + H^2$ regularization In v2, the regularization only penalized strong curvature in the reconstructed potential. While this produced a smoother result, the potential could still contain regions where it changed too abruptly. Version 3 also penalizes large slopes, which makes the potential vary more gradually. Together, the two penalties produce a smoother and more realistic potential that better resembles the expected parabolic shape of the quantum harmonic oscillator.

2D GCV parameter selection The amount of smoothing depends on the regularization parameters. If they are too small, the reconstruction follows the noise; if they are too large, important features may be oversmoothed. Instead of selecting these parameters manually, v3 automatically searches for the best pair (α_1, α_2) using the Generalized Cross-Validation (GCV) criterion. This provides a good balance between noise reduction and reconstruction accuracy without requiring manual tuning.

2D joint smoothing Since the observed wavefunction depends on both position and time, noise is present in both directions. Rather than smoothing each dimension separately, v3 treats the wavefunction as a single two-dimensional surface and applies Gaussian smoothing simultaneously in space and time. This removes noise more naturally while preserving the overall evolution of the wavefunction.

ResNet architecture The neural network is upgraded from a standard multi-layer perceptron (MLP) to a Residual Network (ResNet) [55]. Residual, or skip, connections allow information to flow directly through the network, making training more stable and helping the model learn the rapidly oscillating patterns that naturally appear in quantum wavefunctions.

Causal training The Schrödinger equation describes how a quantum system evolves forward in time from an initial state. Causal training follows this by encouraging the network to first satisfy the equation at earlier times before focusing on later times. It reduces the risk of learning the final dynamics accurately while neglecting the beginning of the evolution.

RAR sampling Not every region of the solution is equally difficult to learn. Instead of using the same collocation points throughout training, v3 periodically identifies where the PDE residual is largest and places more training points in those regions. This focuses the learning process on the areas where the network makes its largest errors.

Curriculum on w_{PDE} At the beginning of training, enforcing the physics too strongly can make optimization unstable. Therefore, v3 gradually increases the weight of the PDE loss during the first iterations. This helps the network to first learn the observed wavefunction and then progressively refine its prediction so that it satisfies the Schrödinger equation more accurately.

5.7.2 Architecture Comparison

Table 5.7: Tikhonov variant configurations.

Parameter	v1	v2	v3
FD stencil	Order 2	Order 4	Order 4
Smoothing	SavGol 1D	SavGol 1D	Gaussian 2D + SavGol
Regularization	H^2	H^2	$H^1 + H^2$
α selection	GCV 1D	GCV 1D	GCV 2D
Weighting	$ \psi $	$ \psi $	$ \psi ^2$

Table 5.8: Inverse PINN variant configurations.

Parameter	v1	v2	v3
ψ_{net} units/depth	48/4	64/4	64/5
V_{net} units/depth	32/3	32/3	48/4
Architecture	MLP tanh	MLP tanh	ResNet tanh
Fourier features	No	Yes (32)	Yes (48, σ auto)
Data points	2500	2500	4000
Collocation points	4000	4000	6000
IC points	200	200	300
Adam iterations	2500	2000	4000
L-BFGS iterations	—	300	500
Norm penalty	No	Yes	Yes
BC loss	No	No	Yes
Causal training	No	No	Yes
RAR sampling	No	No	Yes
Curriculum	No	No	Yes

5.7.3 Results

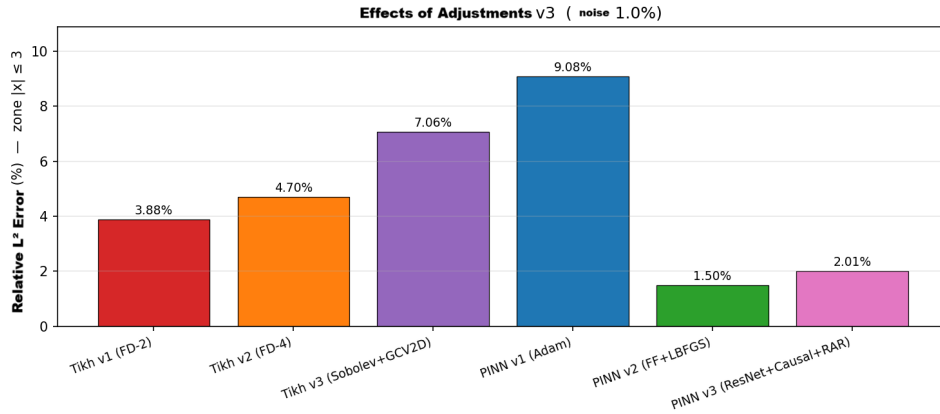


Figure 5.8: Relative L^2 error for all six variants at 1% noise, evaluated on $|x| \leq 3$.

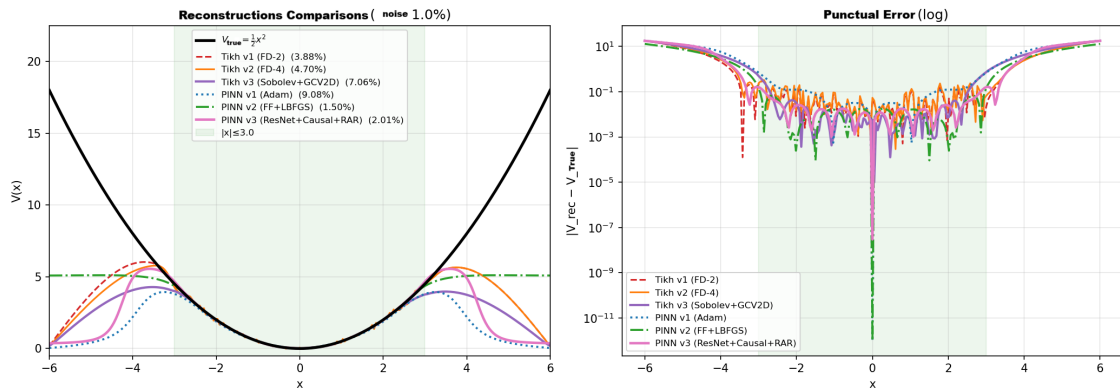


Figure 5.9: All six variants at 1% noise. Left: reconstructed potentials inside $|x| \leq 3$. PINN v2 (green) and Tikhonov v1 (red) track the parabola most closely; Tikhonov v3 (purple) and PINN v3 (pink) show visible deviation. Right: pointwise absolute error on log scale. PINN v2 achieves the lowest and smoothest error profile inside the zone; Tikhonov v3 and PINN v3 are noisier than their respective predecessors.

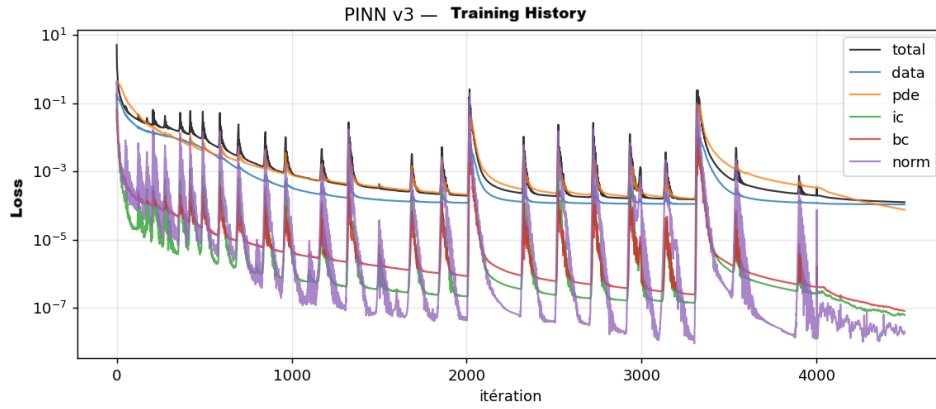


Figure 5.10: PINN v3 training history. The regular large spikes every ~ 300 iterations are RAR resampling events. IC (green) and BC (red) losses reach $\sim 10^{-7}$, confirming constraints are well satisfied. The PDE (orange) and data (blue) losses remain the bottleneck, plateauing around 10^{-3} – 10^{-4} by iteration 4500.

Table 5.9: Full benchmark results at 1% noise, evaluated on $|x| \leq 3$.

Method	Rel. L^2 error	Runtime (s)
Tikh v1 (FD-2)	3.88%	0.32
Tikh v2 (FD-4)	4.70%	0.31
Tikh v3 (Sobolev+GCV2D)	7.06%	1.53
PINN v1 (Adam)	9.08%	114
PINN v2 (FF+LBFGS)	1.50%	205
PINN v3 (ResNet+Causal+RAR)	2.01%	1967

5.7.4 Analysis

The v3 results extend and confirm the counterintuitive pattern already observed at v2: on both sides, adding complexity does not improve accuracy.

Tikhonov: monotone regression with complexity All three Tikhonov variants degrade monotonically, v1 (3.88%), v2 (4.70%), v3 (7.06%) — despite each

adding more sophisticated ingredients. Tikhonov v3 introduces the most powerful regularization framework (Sobolev $H^1 + H^2$, 2D GCV, quadratic weighting) but produces the worst result. The reason is the same as for v2: at 1% noise, the problem is noise-dominated, not truncation-dominated. The 2D GCV search over (α_1, α_2) adds a second regularization axis, but with noise driving the error, optimizing two smoothness penalties simultaneously does not reduce the fundamental noise amplification in the derivatives. Tikhonov v1 the simplest variant, remains the best classical method.

PINN: more complexity, slightly worse result PINN v3 (2.01%) regresses slightly from PINN v2 (1.50%) despite adding ResNet, causal training, RAR, BC loss, and curriculum learning. The training history shows that IC and BC constraints are very well satisfied ($\sim 10^{-7}$), and RAR resampling is active throughout. The bottleneck is the PDE and data losses, which plateau around 10^{-3} – 10^{-4} and do not decrease further even with 4500 iterations. The added architectural complexity of ResNet and causal training appears to slow convergence on this problem without improving the final solution quality. PINN v2 with its simpler MLP and Fourier features found a better minimum faster.

Overall verdict Across all six variants, PINN v2 achieves the best accuracy (1.50%) and Tikhonov v1 the best classical result (3.88%). The consistent finding is that improvement is not monotone with complexity on either side. For Tikhonov, the simpler the stencil and regularization, the better the result at this noise level. For the PINN, the key ingredients were Fourier features and L-BFGS introduced at v2; everything added at v3 brought marginal cost at 1967s nearly $10\times$ slower than v2 for a slightly worse outcome.

5.8 Benchmark v4: Final Variants

5.8.1 New Adjustments

K-fold cross-validation (Tikhonov v4) Instead of GCV, the pair (α_1, α_2) is selected by 5-fold cross-validation. The rows of the linear system $AV = b$ are split into five equal groups; each fold is held out once as a validation set while the remaining four are used to fit V . The average validation error across the five folds identifies the optimal regularization pair. K-fold is in principle less sensitive to noise fluctuations than GCV because it evaluates generalization directly rather than estimating it analytically.

Early stopping (PINN v4) Training is monitored on a held-out validation set (15% of data points). If the validation loss does not improve for 20 consecutive checks (every 50 iterations), training stops and the best weights are restored. This prevents the network from overfitting to the noisy observations after the validation loss has plateaued.

5.8.2 Architecture Comparison

Table 5.10: Tikhonov v3 vs v4.

Parameter	v3	v4
FD stencil	Order 4	Order 4
Smoothing	Gaussian 2D + SavGol	Gaussian 2D + SavGol
Regularization	$H^1 + H^2$	$H^1 + H^2$
α selection	GCV 2D	K-fold CV 2D
Weighting	$ \psi ^2$	$ \psi ^2$

Table 5.11: Inverse PINN v3 vs v4.

Parameter	v3	v4
Architecture	ResNet (64u, 5L)	ResNet (64u, 5L)
Fourier features	Yes (48, σ auto)	Yes (48, σ auto)
Optimizer	Adam + L-BFGS	Adam + L-BFGS
Collocation pts	6000	12000
Validation set	No	Yes (15%)
Early stopping	No	Yes (patience 20)
Precision	float32	bfloat16 (GPU)
Checkpoint saving	No	Yes (best weights)

5.8.3 Results

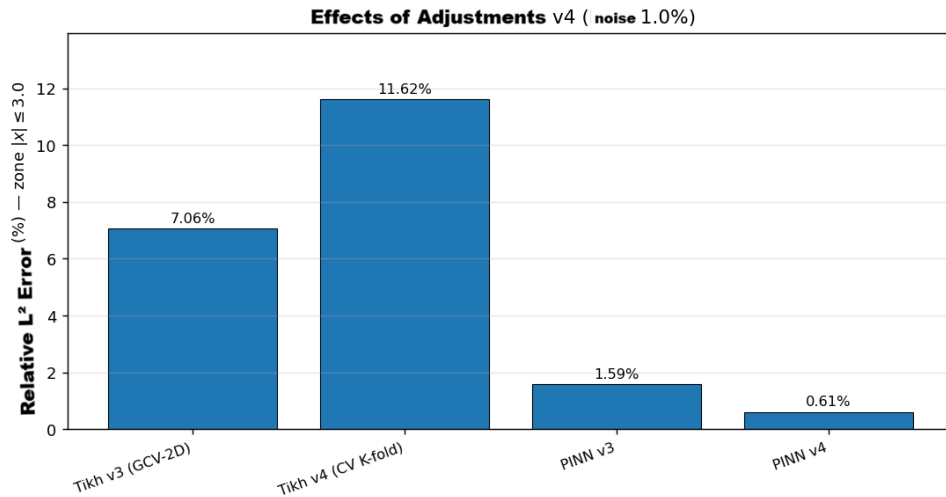


Figure 5.11: Relative L^2 error for Tikhonov v3–v4 and PINN v3–v4 at 1% noise, evaluated on $|x| \leq 3$.

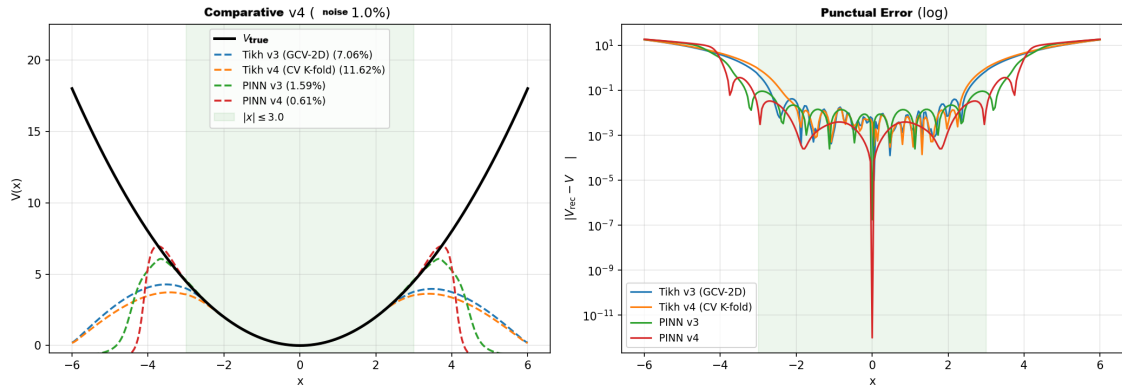


Figure 5.12: Tikhonov v3–v4 vs PINN v3–v4 at 1% noise. Left: reconstructed potentials — PINN v4 (red dashed) tracks the parabola most closely inside $|x| \leq 3$; both Tikhonov variants severely underestimate the potential. Right: pointwise absolute error — PINN v4 achieves the lowest and smoothest error profile, reaching $\sim 10^{-11}$ near $x = 0$.

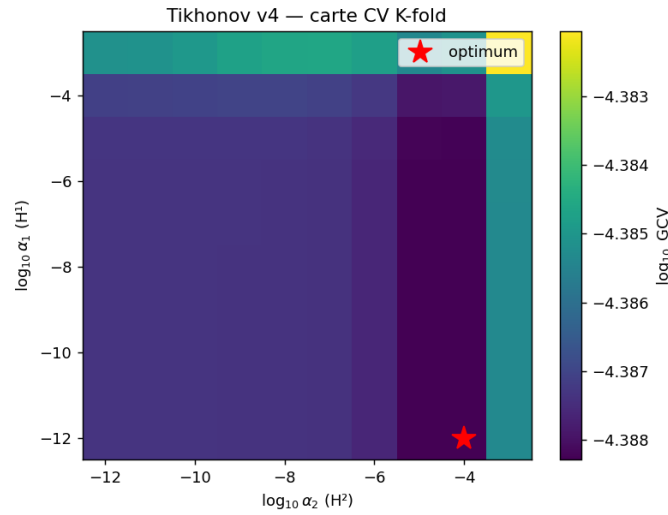


Figure 5.13: K-fold CV score map for Tikhonov v4 over the (α_1, α_2) grid. The optimum (red star) is selected at $\alpha_1 = 10^{-12}$, $\alpha_2 = 10^{-4}$, a corner of the search space indicating that K-fold selected an extreme regularization configuration.

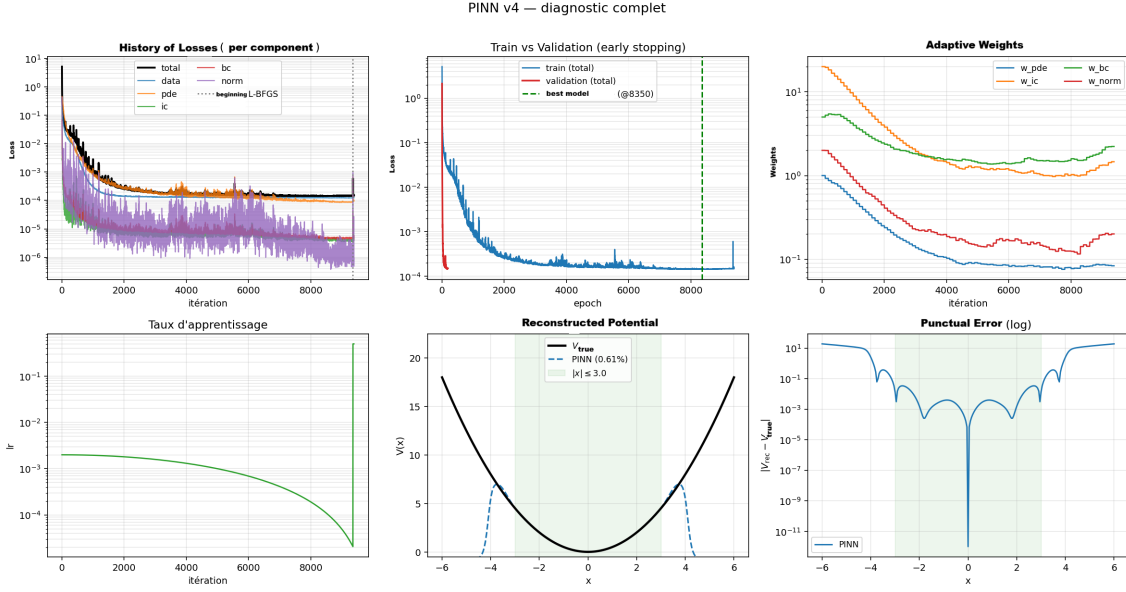


Figure 5.14: PINN v4 diagnostic. Top left: loss components — IC and BC reach $\sim 10^{-6}$; PDE and data plateau around 10^{-4} . Top center: train vs validation loss — best model saved at epoch 8350 before L-BFGS starts. Top right: adaptive weights — w_{IC} decreases as the initial condition is learned; w_{PDE} remains low and stable. Bottom left: learning rate decays smoothly then resets at L-BFGS. Bottom center: reconstructed potential at 0.61% error. Bottom right: pointwise error.

Table 5.12: Benchmark v4 results at 1% noise, evaluated on $|x| \leq 3$.

Method	Rel. L^2 error	Runtime (s)
Tikh v3 (GCV-2D)	7.06%	1.31
Tikh v4 (K-fold)	11.62%	0.27
PINN v3	1.59%	198
PINN v4	0.61%	273

5.8.4 Analysis

Tikhonov v4: K-fold selects a pathological solution Replacing GCV with K-fold worsens the result dramatically (11.62% vs 7.06%). The CV map reveals why: the optimum is located at the corner $(\alpha_1, \alpha_2) = (10^{-12}, 10^{-4})$, meaning K-

fold selected near-zero H^1 regularization paired with moderate H^2 smoothing. This corner solution indicates that the K-fold criterion is not well-suited to this problem at 1% noise: the data rows in $AV = b$ are correlated across time, and splitting them into five independent folds violates the assumption of exchangeability that K-fold requires. GCV, which does not assume independence, is better adapted to this structured data.

PINN v4: early stopping delivers the best result PINN v4 achieves 0.61%, the best result across all eight variants. The key change from v3 is early stopping: training halted at epoch 8350, before the network began overfitting to the noisy observations. The train vs validation plot confirms this, the gap between train and validation loss was minimal at the saved checkpoint, and widened afterward. Doubling the collocation points to 12000 also improved PDE residual coverage. The pointwise error is smooth and reaches $\sim 10^{-11}$ near $x = 0$, the best spatial accuracy observed across all experiments.

5.9 Summary: Inverse Problem

The inverse experiments lead to four clear conclusions.

Both methods recover the potential accurately in the illuminated zone

All variants reconstruct $V(x) = \frac{1}{2}x^2$ well inside $|x| \leq 3$. Outside this zone, where $|\psi| \approx 0$, neither method is constrained by data, Tikhonov oscillates and the PINN collapses to zero. This is a fundamental limitation of the inverse problem itself, not a failure of either algorithm.

More complexity does not always help

On the Tikhonov side, every improvement made things worse: v1 (3.88%) < v2 (4.70%) < v3 (7.06%) < v4 (11.62%). At 1% noise, the problem is noise-dominated: higher-order stencils and richer reg-

ularization amplify noise faster than they reduce truncation error. On the PINN side, the targeted improvements at v2 (Fourier features, L-BFGS, norm penalty) delivered the largest gain; the architectural additions at v3 (ResNet, causal training, RAR) added cost without improving accuracy.

Physical priors are what make the PINN competitive The symmetry constraint and zero-at-origin prior built into V_{net} are what allow the PINN to outperform Tikhonov. Without them, PINN v1 reaches 7.78%–9.08%, worse than every Tikhonov variant. With them and the right training setup, PINN v4 achieves 0.61%, the best result across all experiments. This confirms the theoretical argument of Shumaylov et al. [52]: hard physical priors are mathematically necessary for non-chaotic inverse problems, not just helpful.

Tikhonov wins on speed and generality; PINN wins on accuracy with prior knowledge Tikhonov v1 reconstructs the potential in 0.60 s with no problem-specific knowledge beyond smoothness. PINN v4 achieves 0.61% error but requires 273 s, GPU hardware, and structural knowledge of the potential embedded in the architecture.

Chapter 6

Conclusion and Perspectives

6.1 Conclusion

This thesis addressed the same physical question through two different computational approaches: given the time-dependent Schrödinger equation for the one-dimensional quantum harmonic oscillator, how can the wave function be obtained when the potential is known, and how can the potential be recovered when only noisy observations of the wave function are available. The forward and inverse problems are not independent. The inverse problem is only meaningful once the forward problem is solved, and it is precisely the inverse problem that exposes the limitations a forward solver alone does not reveal.

The classical part of this work discretizes the wave function on a grid and advances it in time through the implicit Crank-Nicolson scheme, the only time-stepping method considered that is simultaneously unconditionally stable and exactly unitary, properties that explicit schemes such as FTCS or leapfrog cannot offer together. Recovering the potential from observed wave functions is ill-posed in the sense of Hadamard: small perturbations in the data can produce large changes in the reconstructed potential. Tikhonov regularization restores a stable and well-posed reformulation by balancing data fidelity against solution smoothness, with

the regularization parameter selected automatically through cross-validation. The structural cost of this approach is what matters most for the comparison that follows: the inverse algorithm is an outer optimization loop wrapped around a forward solver that must be rerun at every iteration, and the entire pipeline scales poorly once the system grows beyond one spatial dimension.

The learning-based alternative developed in this thesis represents the wave function differently, not as values on a mesh, but as the output of a neural network queried at arbitrary coordinates, trained to satisfy the Schrödinger equation as a residual condition enforced through the loss function rather than through a discretized operator. The same architecture is extended to the inverse problem by declaring the unknown potential a trainable parameter alongside the network weights, updated by the same backpropagation pass that updates everything else. The non-uniqueness inherent to the inverse problem reappears here exactly as it did in the classical setting. It is not introduced by the neural network; it is a property of the inverse problem itself. Both frameworks address it through the same idea that constraining the solution space through regularization or prior knowledge until a single physically meaningful minimum remains.

The forward PINN results on the one-dimensional harmonic oscillator were less accurate than Crank-Nicolson, and this outcome deserves careful interpretation rather than a dismissal of the method. The harmonic oscillator is precisely the class of problem where classical solvers excel: a regular domain, a smooth potential, a uniform grid, and a time evolution that is linear and unitary. Crank-Nicolson was designed for this setting and performs exactly as its mathematical guarantees predict. A PINN trained on the same problem faces a different challenge, it must approximate a complex oscillatory space-time function using a neural network optimized through a non-convex loss landscape, without any of the structural advantages the finite-difference grid provides. The fact that the forward PINN does not outperform Crank-Nicolson on a one-dimensional linear problem is not a failure of the

method. It is the expected outcome of applying a general-purpose tool to a problem that a specialized tool already handles optimally. Comparing training time to the runtime of a tridiagonal solver is also misleading because the PINN is not a faster version of Crank-Nicolson, it is a fundamentally different kind of solver built for problems where Crank-Nicolson cannot be applied at all.

The inverse results tell the more important result. On the potential reconstruction benchmark, the best PINN variant achieved a relative L^2 error of 0.61%, outperforming all Tikhonov variants across all configurations tested. This is where the PINN architecture earns its cost: the forward physics and the inverse objective are encoded simultaneously inside a single loss function, there is no outer optimization loop, no forward solver to rerun at every step, and the method extends naturally to higher-dimensional domains where Tikhonov regularization would require a complete reformulation. The computational cost of training reflects the generality of what is being learned, not inefficiency.

The two frameworks are therefore not simply two unrelated methods solving the same equation. Tikhonov regularization requires a fresh forward solve at every optimization step, while a physics-informed neural network is trained once and encodes both the forward physics and the inverse objective inside a single architecture simultaneously. This structural difference is the core argument this thesis is built around, and it is what the results put to the test directly.

What this thesis demonstrates is that the choice between a classical solver and a physics-informed neural network is not a question of which one is unconditionally better, but of which structural trade-offs are acceptable for a given problem. The classical approach offers mathematical guarantees, unitarity, exact probability conservation, and a well-understood convergence theory, at the price of a rigid mesh and a forward-inverse pipeline that must be rebuilt for every new problem. The PINN approach trades these guarantees for flexibility: a single mesh-free framework that handles the forward and inverse problem within the same architecture, naturally

suited to irregular geometries, higher dimensions, and problems where the governing physics is known but classical discretization is intractable. The one-dimensional harmonic oscillator, by design, favors the classical approach. The value of PINNs becomes visible precisely when that design assumption breaks down.

6.2 Perspectives

Several directions suggest themselves as natural extensions of this work:

Reproducibility and open science The code developed for this thesis, covering the Crank–Nicolson solver, the Tikhonov inverse algorithm, and both the forward and inverse physics-informed neural networks, will be organized into a public repository with structured documentation, usage examples, and a self-contained environment specification, intended to serve as both a record of the computational results and a learning resource for students approaching these methods for the first time.

Extension to higher dimensions and more complex systems The one-dimensional harmonic oscillator is a deliberate choice of benchmark, clean enough to isolate the structural comparison between methods, but not representative of realistic quantum problems. The most immediate extension is to two- and three-dimensional potentials, where the curse of dimensionality makes classical grid-based methods increasingly intractable while the mesh-free PINN degrades more gracefully. Multi-particle systems and anharmonic or double-well potentials are also further targets. A particular direction concerns chaotic quantum systems, as established by Shumaylov et al., chaotic systems are more discoverable from data than stable periodic ones, because their trajectories explore the state space densely enough to uniquely constrain the governing dynamics [52], which would structurally reduce the non-uniqueness difficulties encountered here.

Alternative PINN formulations and training strategies The PINN implemented in this thesis follows the original collocation-based formulation with gradient descent. Several alternatives are worth exploring on the Schrödinger problem. Residual-based adaptive weighting, as proposed by Anagnostopoulos et al., dynamically focuses the optimizer on underresolved regions of the domain [53], and has recently been given a rigorous variational justification by Toscano et al. as dual minimization of the maximum rather than average residual [56].

Inverse problems as a research priority The most important long-term direction is the continued development of data-driven methods for quantum inverse problems. Forward simulation is a mature field; inverse problems receive comparatively less attention, despite being closer to what experiment actually provides, observations of a system whose underlying structure must be inferred. Spectroscopy, quantum control, material characterization, and quantum device design are all, at their core, inverse problems. The present moment is particularly favorable. Increasingly rich experimental datasets combined with physics-informed machine learning create conditions in which inverse problems that were previously intractable may become solvable. Extending the framework developed here to higher-dimensional systems, time-dependent potentials, and observable-based measurements connects to some of the most active open questions in computational physics.

Towards neural network potentials and density functional theory A promising long-term direction concerns the connection between the inverse potential recovery framework developed here and the construction of interatomic potentials from quantum mechanical data. The work of Behler and Parrinello established that neural networks can learn high-dimensional potential energy surfaces from ab initio calculations with near-quantum-mechanical accuracy [57]. Both problems share the same fundamental structure: recovering an unknown potential from quantum mechanical observations under sufficient physical constraints. Exploring this con-

nection represents an ambitious continuation of the research direction opened by this thesis.

Bibliography

- [1] David J. Griffiths and Darrell F. Schroeter. *Introduction to Quantum Mechanics*. en. 3rd ed. Cambridge University Press, Aug. 2018. ISBN: 978-1-316-99543-3. DOI: [10.1017/9781316995433](https://doi.org/10.1017/9781316995433). URL: <https://www.cambridge.org/highereducation/product/9781316995433/book> (visited on 03/15/2026).
- [2] J. J. Sakurai and Jim Napolitano. *Modern Quantum Mechanics*. 2nd ed. Cambridge University Press, 2017. DOI: [10.1017/9781108499996](https://doi.org/10.1017/9781108499996).
- [3] John Crank and Phyllis Nicolson. “A Practical Method for Numerical Evaluation of Solutions of Partial Differential Equations of the Heat-Conduction Type”. In: *Mathematical Proceedings of the Cambridge Philosophical Society* 43.1 (1947), pp. 50–67. DOI: [10.1017/S0305004100023197](https://doi.org/10.1017/S0305004100023197).
- [4] Randall J. LeVeque. *Finite Difference Methods for Ordinary and Partial Differential Equations: Steady-State and Time-Dependent Problems*. en. Society for Industrial and Applied Mathematics, Jan. 2007. ISBN: 978-0-89871-629-0. DOI: [10.1137/1.9780898717839](https://doi.org/10.1137/1.9780898717839). URL: <http://epubs.siam.org/doi/book/10.1137/1.9780898717839>.
- [5] P. B. Visscher. “A fast explicit algorithm for the time-dependent Schrödinger equation”. en. In: *Computers in Physics* 5.6 (Nov. 1991), pp. 596–598. DOI: [10.1063/1.168415](https://doi.org/10.1063/1.168415). URL: <https://pubs.aip.org/cip/article/5/6/596/279764/A-fast-explicit-algorithm-for-the-time-dependent>.

- [6] Jacques Hadamard. *Lectures on Cauchy's Problem in Linear Partial Differential Equations*. Yale University Press, 1923.
- [7] Andreas Kirsch. *An Introduction to the Mathematical Theory of Inverse Problems*. en. Vol. 120. Applied Mathematical Sciences. Cham: Springer International Publishing, 2021. ISBN: 978-3-030-63342-4. DOI: [10.1007/978-3-030-63343-1](https://doi.org/10.1007/978-3-030-63343-1). URL: <http://link.springer.com/10.1007/978-3-030-63343-1>.
- [8] Andrey N. Tikhonov and Vasiliy Y. Arsenin. *Solutions of Ill-Posed Problems*. Washington, DC: Winston and Sons, 1977.
- [9] Heinz W. Engl, Martin Hanke-Bourgeois, and Andreas Neubauer. *Regularization of inverse problems*. Mathematics and its applications <Dordrecht> 375. Dordrecht: Kluwer Acad. Publ, 2000. ISBN: 978-0-7923-4157-4.
- [10] Isaac E. Lagaris, Aristidis Likas, and Dimitrios I. Fotiadis. “Artificial Neural Networks for Solving Ordinary and Partial Differential Equations”. In: *IEEE Transactions on Neural Networks* 9.5 (1998), pp. 987–1000. DOI: [10.1109/72.712178](https://doi.org/10.1109/72.712178).
- [11] Maziar Raissi, Paris Perdikaris, and George E. Karniadakis. “Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations”. en. In: *Journal of Computational Physics* 378 (Feb. 2019), pp. 686–707. DOI: [10.1016/j.jcp.2018.10.045](https://doi.org/10.1016/j.jcp.2018.10.045). URL: <https://linkinghub.elsevier.com/retrieve/pii/S0021999118307125>.
- [12] Salvatore Cuomo et al. “Scientific Machine Learning through Physics-Informed Neural Networks: Where We Are and What’s Next”. In: *Journal of Scientific Computing* 92.3 (2022), p. 88. DOI: [10.1007/s10915-022-01939-z](https://doi.org/10.1007/s10915-022-01939-z).
- [13] Joshua Izaac and Jingbo Wang. *Computational Quantum Mechanics*. en. Undergraduate Lecture Notes in Physics. Cham: Springer International Publish-

- ing, 2018. ISBN: 978-3-319-99930-2. DOI: [10.1007/978-3-319-99930-2](https://doi.org/10.1007/978-3-319-99930-2). URL: <http://link.springer.com/10.1007/978-3-319-99930-2>.
- [14] Max Born. “Zur Quantenmechanik der Stoßvorgänge”. de. In: *Zeitschrift für Physik* 37.12 (Dec. 1926), pp. 863–867. DOI: [10.1007/BF01397477](https://doi.org/10.1007/BF01397477). URL: <http://link.springer.com/10.1007/BF01397477>.
- [15] Ramamurti Shankar. *Principles of quantum mechanics*. 2nd ed. New York: Plenum Press, 1994. ISBN: 978-0-306-44790-7.
- [16] W. Heisenberg. “Über den anschaulichen Inhalt der quantentheoretischen Kinematik und Mechanik”. In: *Zeitschrift für Physik* 43.3 (Mar. 1927), pp. 172–198. DOI: [10.1007/BF01397280](https://doi.org/10.1007/BF01397280). URL: <https://doi.org/10.1007/BF01397280>.
- [17] Michael Reed and Barry Simon. *Methods of Modern Mathematical Physics, Volume I: Functional Analysis*. Academic Press, 1972. ISBN: 9780125850018.
- [18] Jacques Hadamard. “Sur les problèmes aux dérivées partielles et leur signification physique”. In: *Bulletin de la Société Mathématique de France* 30 (1902), pp. 155–200.
- [19] U. M. Ascher and Linda Ruth Petzold. *Computer methods for ordinary differential equations and differential-algebraic equations*. Philadelphia: Society for Industrial and Applied Mathematics, 1998. ISBN: 978-0-89871-412-8.
- [20] I. V. Mel’nikova. “Regularization of ill-posed differential problems”. In: *Siberian Mathematical Journal* 33.2 (Mar. 1992), pp. 289–298. DOI: [10.1007/BF00971100](https://doi.org/10.1007/BF00971100). URL: <https://doi.org/10.1007/BF00971100>.
- [21] Per Christian Hansen. “Analysis of Discrete Ill-Posed Problems by Means of the L-Curve”. en. In: *SIAM Review* 34.4 (Dec. 1992), pp. 561–580. DOI: [10.1137/1034115](https://doi.org/10.1137/1034115). URL: <http://epubs.siam.org/doi/10.1137/1034115>.

- [22] Andrea Amalia Almasy. “Inverse problems in classical and quantum physics”. en. PhD thesis. Johannes Gutenberg-Universität Mainz, 2007. DOI: [10.25358/OPENSOURCE-3346](https://doi.org/10.25358/OPENSOURCE-3346). URL: <https://openscience.ub.uni-mainz.de/handle/20.500.12030/3348> (visited on 03/15/2026).
- [23] V. A. Morozov. “Regular Methods for Solving Linear and Nonlinear Ill-Posed Problems”. en. In: *Methods for Solving Incorrectly Posed Problems*. New York, NY: Springer New York, 1984, pp. 65–122. ISBN: 978-1-4612-5280-1. DOI: [10.1007/978-1-4612-5280-1_3](https://doi.org/10.1007/978-1-4612-5280-1_3). URL: http://link.springer.com/10.1007/978-1-4612-5280-1_3.
- [24] Konstantin Eduardovich Plokhovnikov. “Solving the Schrodinger Equation on the Basis of Finite-Difference and Monte-Carlo Approaches”. en. In: *Journal of Applied Mathematics and Physics* 09.02 (2021), pp. 328–369. DOI: [10.4236/jamp.2021.92024](https://doi.org/10.4236/jamp.2021.92024). URL: <https://www.scirp.org/journal/doi.aspx?doi=10.4236/jamp.2021.92024>.
- [25] Richard L. Burden, John Douglas Faires, and Annette M. Burden. *Numerical analysis*. Tenth edition. Boston, MA: Cengage Learning, 2016. ISBN: 978-1-305-25366-7.
- [26] Jule G. Charney, Ragnar Fjørtoft, and John von Neumann. “Numerical Integration of the Barotropic Vorticity Equation”. In: *Tellus* 2.4 (1950), pp. 237–254. DOI: [10.3402/tellusa.v2i4.8607](https://doi.org/10.3402/tellusa.v2i4.8607).
- [27] John C. Butcher. *Numerical Methods for Ordinary Differential Equations*. 2nd. Chichester, UK: John Wiley & Sons, 2008. DOI: [10.1002/9780470753767](https://doi.org/10.1002/9780470753767).
- [28] Richard Courant, Kurt Friedrichs, and Hans Lewy. “Über die partiellen Differenzengleichungen der mathematischen Physik”. In: *Mathematische Annalen* 100.1 (1928), pp. 32–74. DOI: [10.1007/BF01448839](https://doi.org/10.1007/BF01448839).
- [29] Richard Bellman. *Dynamic Programming*. Princeton, NJ: Princeton University Press, 1957.

- [30] A N Karapetyants, K V Khmelnytskaya, and V V Kravchenko. “A Practical Method for Solving the Inverse Quantum Scattering Problem on a Half Line”. en. In: *Journal of Physics: Conference Series* 1540.1 (Apr. 2020), p. 012007. DOI: [10.1088/1742-6596/1540/1/012007](https://doi.org/10.1088/1742-6596/1540/1/012007). URL: <https://iopscience.iop.org/article/10.1088/1742-6596/1540/1/012007>.
- [31] Vladislav V. Kravchenko. *Reconstruction techniques for inverse Sturm-Liouville problems with complex coefficients*. en. arXiv:2506.00670 [math.CA]. May 2025. DOI: [10.48550/arXiv.2506.00670](https://doi.org/10.48550/arXiv.2506.00670). URL: <http://arxiv.org/abs/2506.00670>.
- [32] Roger Penrose. “A Generalized Inverse for Matrices”. In: *Mathematical Proceedings of the Cambridge Philosophical Society* 51.3 (1955), pp. 406–413. DOI: [10.1017/S0305004100030401](https://doi.org/10.1017/S0305004100030401).
- [33] Christian Clason. *Regularization of Inverse Problems*. en. arXiv:2001.00617 [math.FA]. Feb. 2021. DOI: [10.48550/arXiv.2001.00617](https://doi.org/10.48550/arXiv.2001.00617). URL: <http://arxiv.org/abs/2001.00617>.
- [34] Pauli Virtanen et al. “SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python”. In: *Nature Methods* 17.3 (2020), pp. 261–272. DOI: [10.1038/s41592-019-0686-2](https://doi.org/10.1038/s41592-019-0686-2).
- [35] Gene H. Golub, Michael Heath, and Grace Wahba. “Generalized Cross-Validation as a Method for Choosing a Good Ridge Parameter”. In: *Technometrics* 21.2 (1979), pp. 215–223. DOI: [10.1080/00401706.1979.10489751](https://doi.org/10.1080/00401706.1979.10489751).
- [36] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. Cambridge, MA: MIT Press, 2016. URL: <https://www.deeplearningbook.org>.
- [37] Stuart Geman, Elie Bienenstock, and René Doursat. “Neural Networks and the Bias/Variance Dilemma”. In: *Neural Computation* 4.1 (1992), pp. 1–58. DOI: [10.1162/neco.1992.4.1.1](https://doi.org/10.1162/neco.1992.4.1.1).

- [38] Adam Paszke et al. “PyTorch: An Imperative Style, High-Performance Deep Learning Library”. In: *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035.
- [39] Warren S. McCulloch and Walter Pitts. “A Logical Calculus of the Ideas Immanent in Nervous Activity”. In: *Bulletin of Mathematical Biophysics* 5 (1943), pp. 115–133. DOI: [10.1007/BF02478259](https://doi.org/10.1007/BF02478259).
- [40] Frank Rosenblatt. “The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain”. In: *Psychological Review* 65.6 (1958), pp. 386–408. DOI: [10.1037/h0042519](https://doi.org/10.1037/h0042519).
- [41] George Cybenko. “Approximation by Superpositions of a Sigmoidal Function”. In: *Mathematics of Control, Signals and Systems* 2.4 (1989), pp. 303–314. DOI: [10.1007/BF02551274](https://doi.org/10.1007/BF02551274).
- [42] Kurt Hornik. “Approximation Capabilities of Multilayer Feedforward Networks”. In: *Neural Networks* 4.2 (1991), pp. 251–257. DOI: [10.1016/0893-6080\(91\)90009-T](https://doi.org/10.1016/0893-6080(91)90009-T).
- [43] Vinod Nair and Geoffrey E. Hinton. “Rectified Linear Units Improve Restricted Boltzmann Machines”. In: *Proceedings of the 27th International Conference on Machine Learning (ICML)*. Haifa, Israel, 2010, pp. 807–814.
- [44] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. “Deep Sparse Rectifier Neural Networks”. In: *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS)*. Vol. 15. Proceedings of Machine Learning Research. 2011, pp. 315–323.
- [45] Vincent Sitzmann et al. “Implicit Neural Representations with Periodic Activation Functions”. In: *Advances in Neural Information Processing Systems*. 2020.

- [46] Matthew Tancik et al. “Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains”. In: *Advances in Neural Information Processing Systems*. 2020.
- [47] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. “Learning Representations by Back-Propagating Errors”. In: *Nature* 323.6088 (1986), pp. 533–536. DOI: [10.1038/323533a0](https://doi.org/10.1038/323533a0).
- [48] Atilim Gunes Baydin et al. “Automatic Differentiation in Machine Learning: A Survey”. In: *Journal of Machine Learning Research* 18.153 (2018), pp. 1–43.
- [49] Augustin-Louis Cauchy. “Méthode générale pour la résolution des systèmes d’équations simultanées”. In: *Comptes Rendus de l’Académie des Sciences* 25 (1847), pp. 536–538.
- [50] Ehsan Kharazmi, Zhongqiang Zhang, and George Em Karniadakis. “hp-VPINNs: Variational Physics-Informed Neural Networks with Domain Decomposition”. In: *Computer Methods in Applied Mechanics and Engineering* 374 (2021), p. 113547.
- [51] Xiaowei Jin et al. “NSFnets (Navier-Stokes Flow Nets): Physics-Informed Neural Networks for the Incompressible Navier-Stokes Equations”. In: *Journal of Computational Physics* 426 (2021), p. 109951.
- [52] Zakhar Shumaylov et al. “When Is a System Discoverable from Data? Discovery Requires Chaos”. In: *arXiv preprint arXiv:2511.08860* (2025). arXiv: [2511.08860](https://arxiv.org/abs/2511.08860) [math.DS].
- [53] Sokratis J. Anagnostopoulos et al. “Learning in PINNs: Phase transition, diffusion equilibrium, and generalization”. In: *Neural Networks* 193 (2026), p. 107983. DOI: [10.1016/j.neunet.2025.107983](https://doi.org/10.1016/j.neunet.2025.107983).
- [54] Abraham Savitzky and Marcel J. E. Golay. “Smoothing and Differentiation of Data by Simplified Least Squares Procedures”. In: *Analytical Chemistry* 36.8 (1964), pp. 1627–1639. DOI: [10.1021/ac60214a047](https://doi.org/10.1021/ac60214a047).

- [55] Kaiming He et al. “Deep Residual Learning for Image Recognition”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 770–778. DOI: [10.1109/CVPR.2016.90](https://doi.org/10.1109/CVPR.2016.90).
- [56] Juan Diego Toscano et al. “A Variational Framework for Residual-Based Adaptivity in Neural PDE Solvers and Operator Learning”. In: *arXiv preprint arXiv:2509.14198* (2025). arXiv: [2509.14198](https://arxiv.org/abs/2509.14198) [cs.LG].
- [57] Jörg Behler and Michele Parrinello. “Generalized Neural-Network Representation of High-Dimensional Potential-Energy Surfaces”. In: *Physical Review Letters* 98 (2007), p. 146401. DOI: [10.1103/PhysRevLett.98.146401](https://doi.org/10.1103/PhysRevLett.98.146401).

Résumé

Comment les réseaux de neurones informés par la physique (PINNs) se comportent-ils face aux méthodes numériques traditionnelles sur l'équation de Schrödinger 1D ? Pour le savoir, nous avons étudié un oscillateur harmonique quantique à travers des problèmes directs et inverses. Pour la partie directe, le schéma de Crank-Nicolson et la méthode des valeurs propres nous ont servi de référence. Pour la partie inverse, nous avons testé la régularisation de Tikhonov contre plusieurs variantes de PINNs. Il en ressort que les approches classiques gagnent sur l'efficacité et la résistance au bruit. Toutefois, les PINNs prennent l'avantage en précision, mais seulement si l'on intègre intelligemment les lois physiques dès la conception du réseau.

Mots-clés : Réseaux de neurones informés par la physique, équation de Schrödinger, oscillateur harmonique quantique, problème inverse, régularisation de Tikhonov, méthodes numériques.

Abstract

We compared physics-informed neural networks (PINNs) with traditional numerical approaches to solve forward and inverse problems involving the 1D time-dependent Schrödinger equation for a quantum harmonic oscillator. Classical methods-specifically the Crank-Nicolson scheme and the eigenvalue method-served as our benchmarks for the forward problem. For the inverse problem, we tested Tikhonov regularization against different versions of the inverse PINN. Our findings show that while classical methods are still faster and more resilient to noise, PINNs have a clear edge in accuracy, provided you embed the right physical constraints straight into the network's architecture.

Keywords: Physics-informed neural networks, Schrödinger equation, quantum harmonic oscillator, inverse problem, Tikhonov regularization, numerical methods.

ملخص

تتم في هذا العمل مقارنة شبكات الأعصاب المُخبرة بالفيزياء (PINNs) مع الأساليب العددية الكلاسيكية لحل المسائل المباشرة والعكسية المرتبطة بمعادلة شرودينغر أحادية البعد المعتمدة على الزمن، في حالة المتذبذب التوافقي الكمي. بالنسبة للمسألة المباشرة، يُستخدم مخطط كرانك-نيكولسون وطريقة القيم الذاتية كمراجع. أما للمسألة العكسية، فيُقارن التنظيم وفق تيخونوف مع شبكة PINN العكسية عبر متغيرات متعددة. تُظهر النتائج أن الأساليب الكلاسيكية تبقى متفوقة من حيث الكفاءة والمتانة أمام الضوضاء، في حين تحقق الشبكات دقة أعلى عند دمج القيود الفيزيائية المناسبة مباشرة في البنية المعمارية للشبكة.

الكلمات المفتاحية: شبكات الأعصاب المُخبرة بالفيزياء، معادلة شرودينغر، المتذبذب التوافقي الكمي، المسألة العكسية، تنظيم تيخونوف، الأساليب العددية.