Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

UNIVERSITÀ
DELLA CALABRIA

Master in Mathematics
Dual degree Program

Quantum Algorithms for PDEs: Leveraging Schrödingerization

Giacomo Lorelli

ID 3301039

Supervisor

Prof. Christian Klingenberg
University of Würzburg

Co-supervisors

Prof. Francesco Dell'accio
Università della Calabria

Prof.ssa Filomena Di Tommaso
Università della Calabria

Academic Year 2025/2026

* LaTeX document compiled on July 16, 2026.

Additional resources

Please note that all the code discussed in this work has been made available by the articles' authors at the following repository.

hjp3268/Schrodingerisation-Circuits

To my parents
and friends

Abstract

In 1994, Peter Shor proposed an algorithm for finding the prime factors of an integer that paved the way for an emerging paradigm: quantum computing algorithms could offer substantial speedups compared to their classical counterparts. Within this framework, one research direction focuses on the numerical simulation of partial differential equations (PDEs), aiming to mitigate the so-called "curse of dimensionality". A major bottleneck for classical algorithms that triggers an exponential growth in computational effort as the spatial dimension increases.

The primary tool driving quantum algorithms is Hamiltonian simulation which, however, can only be directly applied to dynamical systems governed by Hermitian operators (such as the Schrödinger equation). This restriction severely limits its applicability to the vast class of PDEs of physical and engineering interest, which are frequently described by non-Hermitian operators (e.g., diffusion, advection, or more generally, dissipative systems).

To overcome this limitation, this thesis introduces Schrödingerization, a framework that converts a generic linear PDE into a Schrödinger-type equation via the so-called "warped transformation". By introducing an auxiliary variable, this technique maps the problem into a higher dimension and symmetrizes the evolution operator, making it suitable for Hamiltonian simulation. To ensure the work remains self-contained and accessible, the first part introduces the fundamental conceptual tools of quantum computing: qubits, quantum gates, circuits, and measurements.

Although the theoretical foundation of Schrödingerization had already been established, an explicit implementation of these algorithms at the quantum circuit level was still lacking. The central contribution of this thesis lies precisely in the detailed construction of such circuits, following the approach proposed by Hu, Jin, Liu, and Zhang (2024). Starting from the discretization of the operator, the circuits for both the heat equation and the discretized advection equation are explicitly derived, including the encoding of the auxiliary dimension required for the operator transformation.

Furthermore, a rigorous computational complexity analysis is conducted to quantify the number of qubits and elementary operations required. Finally, the practical validity of the proposed circuits is verified through numerical simulations (via Qiskit, utilizing the Statevector, Sampler, and Estimator primitives), demonstrating the accurate reproduction of the expected dynamics for both equations.

The results successfully validate the practical feasibility of Schrödingerization as a general bridge between classical PDEs and quantum Hamiltonian simulation, laying the groundwork for future extensions to non-linear equations and higher-order discretization schemes.

Abstrakt

Im Jahr 1994 schlug Peter Shor einen Algorithmus zur Bestimmung der Primfaktoren einer ganzen Zahl vor, der den Weg für ein neues Paradigma ebnete: Quantenalgorithmen könnten im Vergleich zu ihren klassischen Gegenstücken erhebliche Beschleunigungen bieten. In diesem Rahmen konzentriert sich eine Forschungsrichtung auf die numerische Simulation partieller Differentialgleichungen (PDEs) mit dem Ziel, den sogenannten „Fluch der Dimensionalität“ abzumildern – ein zentrales Hindernis für klassische Algorithmen, das mit zunehmender räumlicher Dimension zu einem exponentiellen Anstieg des Rechenaufwands führt.

Das wichtigste Werkzeug hinter Quantenalgorithmen ist die Hamiltonsche Simulation, die jedoch nur direkt auf dynamische Systeme angewendet werden kann, die durch hermitesche Operatoren beschrieben werden (wie etwa die Schrödinger-Gleichung). Diese Einschränkung begrenzt ihre Anwendbarkeit erheblich auf die große Klasse physikalisch und ingenieurwissenschaftlich relevanter PDEs, die häufig durch nicht-hermitesche Operatoren beschrieben werden (z. B. Diffusion, Advektion oder allgemeiner dissipative Systeme).

Um diese Einschränkung zu überwinden, führt diese Arbeit die Schrödingerisierung ein, einen Rahmen, der eine allgemeine lineare PDE mittels der sogenannten „verzerrten Transformation“ (*warped transformation*) in eine Gleichung vom Schrödinger-Typ überführt. Durch die Einführung einer Hilfsvariablen bildet diese Technik das Problem auf eine höhere Dimension ab und symmetrisiert den Evolutionsoperator, wodurch dieser für die Hamiltonsche Simulation geeignet wird. Damit die Arbeit in sich geschlossen und zugänglich bleibt, führt der erste Teil die grundlegenden konzeptionellen Werkzeuge des Quantencomputings ein: Qubits, Quantengatter, Schaltkreise und Messungen.

Obwohl die theoretische Grundlage der Schrödingerisierung bereits etabliert war, fehlte bislang eine explizite Umsetzung dieser Algorithmen auf der Ebene von Quantenschaltkreisen. Der zentrale Beitrag dieser Arbeit liegt eben in der detaillierten Konstruktion solcher Schaltkreise, in Anlehnung an den von Hu, Jin, Liu und Zhang (2024) vorgeschlagenen Ansatz. Ausgehend von der Diskretisierung des Operators werden die Schaltkreise sowohl für die Wärmeleitungsgleichung als auch für die diskretisierte Advektionsgleichung explizit hergeleitet, einschließlich der Kodierung der für die Operatortransformation erforderlichen Hilfsdimension.

Darüber hinaus wird eine rigorose Analyse der Rechenkomplexität durchgeführt, um die benötigte Anzahl an Qubits und elementaren Operationen zu quantifizieren. Abschließend wird die praktische Gültigkeit der vorgeschlagenen Schaltkreise durch numerische Simulationen (mittels Qiskit, unter Verwendung der Primitiven Statevector, Sampler und Estimator) überprüft, wobei die genaue Reproduktion der erwarteten Dynamik für beide Gleichungen nachgewiesen wird.

Die Ergebnisse bestätigen erfolgreich die praktische Durchführbarkeit der Schrödingerisierung als allgemeine Brücke zwischen klassischen PDEs und der quantenmechanischen Hamiltonschen Simulation und legen damit den Grundstein für zukünftige Erweiterungen auf nichtlineare Gleichungen und Diskretisierungsschemata höherer Ordnung.

Sommario

Nel 1994 l'algoritmo di Shor per la ricerca dei fattori primi di un intero apriva concretamente la strada ad un'idea che stava sviluppandosi in quegli anni, ovvero che algoritmi di computazione quantistica potessero offrire notevoli incrementi nelle velocità computazionali se paragonati agli algoritmi classici. In questo contesto tra le varie direzioni in cui è avanzata la ricerca, una è quella relativa alla simulazione numericha di equazioni alle derivate parziali (PDE), al fine di sopperire alla cosiddetta "maledizione della dimensionalità", dovuta al crescere della dimensione spaziale del problema, di cui gli algoritmi classici soffrono.
Il principale strumento a disposizione degli algoritmi quantistici è la simulazione Hamiltoniana, la quale, tuttavia, può essere applicata direttamento solo a sistemi dinamici governati da operatori Hermitiani (il caso dell'equazione di Schrödinger). Questa restrizione ne limita fortemente l'applicabilità alla vasta classe di PDE di interesse fisico e ingegneristico, spesso descritte da operatori non Hermitiani (equazioni di diffusione, di avvezione, o più in generale sistemi dissipativi). Per superare questo limite, il seguente lavoro di tesi introduce la Schrödingerizzazione (Schrödingerisation), costruzione che permette di convertire una generica PDE lineare in un'equazione di tipo Schrödinger attraverso la cosiddetta "warped transformation": l'aggiunta di una variabile ausiliaria innalza la dimensionalità del problema e simmetrizza l'operatore di evoluzione, rendendolo idoneo alla simulazione Hamiltoniana.
Con lo scopo di rendere più fruibile il lavoro vengono introdotti, in una prima parte, gli strumenti concettuali della computazione quantistica necessari alla trattazione: qubit, gate quantistici, circuiti e misurazioni.
Sebbene il formalismo della Schrödingerizzazione fosse già stato sviluppato a livello teorico, mancava ancora un'implementazione esplicita, a livello di circuito quantistico, degli algoritmi. Il contributo centrale della tesi consiste appunto nella costruzione dettagliata di tali circuiti, seguendo l'approccio proposto da Hu, Jin, Liu e Zhang (2024): a partire dalla discretizzazione dell'operatore, si ricavano esplicitamente i circuiti per l'equazione del calore e per l'equazione di avvezione discretizzata, comprensivi della codifica della dimensione ausiliaria introdotta al fine di poter trasformare l'operatore. Viene inoltre condotta un'analisi di complessità computazionale che quantifica il numero di qubit e di operazioni elementari richieste. Infine, la validità pratica dei circuiti proposti viene verificata attraverso esperimenti numerici su simulatore (tramite Qiskit, utilizzando le primitive Statevector, Sampler ed Estimator), che assicurano la corretta riproduzione della dinamica attesa per entrambe le equazioni considerate.
I risultati ottenuti confermano dunque la fattibilità pratica della Schrödingerizzazione come ponte generale tra PDE classiche e simulazione Hamiltoniana quantistica, aprendo la strada a future estensioni a equazioni non lineari e a schemi di discretizzazione di ordine superiore.

Acknowledgments

I would like to express my sincere gratitude to:

Professor Christian Klingenberg or his guidance throughout the writing of this thesis. His suggestions, starting from recommending the research article that inspired this work, provided the foundation for this work. I am also deeply grateful for the opportunity he gave me by allowing me to attend the weekly meetings with his research group.

Professor Francesco Dell'Accio and Professor Filomena Di Tommaso for their supervision from Italy. Their availability, encouragement, and feedback throughout my university career has been invaluable. Their constant support and confidence in my work have played a fundamental role in my academic development, and I am sincerely grateful for their mentorship.

Professor Francesco De Anna and Professor Concettina Galati for coordinating and supporting the Dual Degree program. Their commitment and dedication made this international academic experience possible, providing me with an invaluable opportunity for both academic and personal development.

Professor Lei Zhang for his willingness to answer my questions and clarify aspects of the Schrödingerization framework proved invaluable. The articles he co-authored was the cornerstone of this work, and his patient correspondence helped me navigate technical difficulties I encountered along the way.

Professor Shi Jin and the members of his research group for their foundational work on Schrödingerization, which this thesis relies upon extensively.

Contents

1 Introduction

The idea of harnessing quantum mechanics to simulate physical systems dates back to Richard Feynman's seminal observation in the 1980s that classical computers appear to incur an exponential cost in simulating quantum dynamics.So it is a truth universally acknowledged, that everything written about quantum simulation must begin with the quote

"[. . .] *nature isn't classical, dammit, and if you want to make a simulation of nature, you'd better make it quantum mechanical* [. . .]" [8]

This insight was later formalized by Seth Lloyd, who provided the first explicit quantum algorithm for simulating local Hamiltonians [18], establishing quantum simulation as one of the most natural and promising applications of quantum computing. In the decades that followed, tremendous progress was made in refining Hamiltonian simulation techniques. Early algorithms by Aharonov and Ta-Shma extended the scope to sparse Hamiltonians [1], while subsequent works by Berry and collaborators achieved near-optimal query complexity with respect to the evolution time, sparsity, and precision [2]. These developments cemented Hamiltonian simulation as a cornerstone of quantum algorithm design, with complexity scaling that offers exponential improvements over classical methods in certain regimes. Despite these advances, the vast majority of quantum simulation algorithms are designed for systems governed by the Schrödinger equation, that is, unitary dynamics generated by a Hermitian Hamiltonian. However, most problems in science and engineering are modelled by partial differential equations (PDEs) that do not possess an underlying Schrödinger structure. For example the heat equation, advection–diffusion equations, and more general dissipative or hyperbolic systems fall into this category.

In classical computation, such equations are typically discretized using finite-difference, finite-element, hybrid or spectral methods, reducing the PDE to a system of ordinary differential equations (ODEs) which are then integrated in time. While these classical techniques are mature and highly optimized, they suffer from the so-called curse of dimensionality: the number of grid points grows exponentially with the spatial dimension d, and handling these large-scale simulations becomes prohibitively expensive even on modern high-performance computing clusters. The quest for quantum speedups in solving classical PDEs has motivated two principal algorithmic strategies. The first approach discretizes the PDE in both space and time to obtain a large linear system, which is

then solved using quantum linear systems algorithms such as the HHL algorithm [9] and its improvements [5] as presented in several different articles [7, 19, 24]. These methods can offer polynomial or even super-polynomial speedups over classical direct solvers, however, the output of HHL-based methods is a quantum state $|x\rangle$ proportional to the solution, necessitating very resource-intensive quantum tomography to determine the complete solution x. Moreover, HHL-based methods require extensive quantum resources, such as qubits and gates, making practical application challenging in the era of NISQ (Noisy Intermediate-Scale Quantum) computers.

The second approach, more directly aligned with the spirit of Hamiltonian simulation, seeks to encode the time evolution of the PDE into a quantum state $|u(t)\rangle$ and evolve it via a sequence of quantum gates. This strategy is conceptually elegant but encounters a fundamental obstacle: the semi-discrete operators of dissipative or non-self-adjoint PDEs, such as the Laplacian matrix for the heat equation or the upwind-differenced advection operator, are not Hermitian, and therefore do not generate unitary dynamics amenable to standard Hamiltonian simulation. To circumvent this limitation, the quantum computing team at Institute of Natural Sciences, Shanghai Jiao Tong University in 2022 introduced a framework called *Schrödingerization* [12, 13]. The key insight is to transform a general linear PDE into a Schrödinger type equation in one higher dimension by means of a warped phase transformation. Concretely, for a PDE

$$\frac{\partial u}{\partial t} = Au$$

with non-Hermitian A , one introduces an auxiliary variable p and defines

$$v(t, x, p) = e^{-p}\, u(t, x) \qquad \text{for } p \geq 0.$$

Extending this to $p < 0$ and applying a Fourier transform in the p variable yields a wave function $\hat{v}(t, x, p)$ that evolves under a Hermitian Hamiltonian H , thus restoring unitarity and enabling the use of Hamiltonian simulation algorithms. This technique is remarkably general: it applies to arbitrary linear ODEs and PDEs, including parabolic and hyperbolic problems, and has been extended to iterative linear algebra solvers and reacting flows .

While Schrödingerization provides a theoretical bridge from general linear PDEs to quantum simulation, the explicit design of quantum circuits implementing this transformation remained largely open prior to the work [14]. In particular, earlier studies established the mathematical foundations and asymptotic complexity bounds, but did not furnish gate-level constructions that could be executed on a quantum computer. A notable exception is the recent work by Sato et al. [21], which proposed scalable quantum circuits for wave and Schrödinger type equations using the Bell basis; however, their approach relies on the central difference scheme and is tailored to equations that are already Hermitian or skew-Hermitian, making it unsuitable for genuinely non-unitary dynamics such as the heat equation or the upwind discretization of the advection equation.

The article that forms the centerpiece of this thesis addresses precisely this gap. The authors present a complete, explicit quantum circuit implementation of the Schrödinger-

ization method for general linear PDEs, with detailed constructions for two paradigmatic examples: the heat equation with Dirichlet boundary conditions and the advection equation with periodic boundary conditions discretized by the upwind scheme. For the heat equation, the circuit implements the time evolution operator $U_{heat} = \exp(iH_{heat}\tau)$ via a first-order Lie–Trotter–Suzuki decomposition, decomposing the shift operators into sequences of CNOT, Hadamard, phase, and multi-controlled rotation gates. Crucially, the upwind scheme, chosen for its stability and absence of spurious oscillations across discontinuities, introduces a dissipative structure that cannot be simulated directly, making the Schrödingerization step essential. Finally, the validity of the proposed circuits is confirmed numerically using the Qiskit simulator, with statevector and shot-based measurements showing close agreement with classical reference solutions obtained from matrix exponentials and from direct Schrödingerization implementations.

1.1 Structure of the Thesis

The remainder of the thesis is organized as follows.

Chapter 2 — Background. We introduce the quantum computational prerequisites. In particular we discuss the gate-based model of quantum computation, Hamiltonian simulation via product formulas, the quantum Fourier transform.
We suggest to readers wishing to gain a more comprehensive understanding of the subject read the manual written by Nielsen and Chuang [20], as it is considered a cornerstone of the field.

Chapter 3 — Schrödingerisation. We develop the general theory of the warped phase transformation in full mathematical rigor, establishing the Schrödinger equation in the extended space, the spectral convergence of the p-discretisation, and the abstract quantum pipeline (state preparation $\rightarrow$ QFT $\rightarrow$ Hamiltonian simulation $\rightarrow$ inverse QFT $\rightarrow$ post-selection).

Chapter 4 — Heat equation. We specialize the framework to the d-dimensional heat equation with Dirichlet boundary conditions, deriving the Hamiltonian H_{heat} and constructing the full circuit hierarchy $W_j \rightarrow V_0 \rightarrow \tilde{V}_0 \rightarrow V_{\text{heat}}$ with complete gate sequences. The Trotter error analysis is carried out in detail, culminating in the one-step and accumulated error bounds.

Chapter 5 — Advection equation. We treat the discretised advection equation, explaining the necessity of Schrödingerisation, deriving the Hermitian decomposition of the upwind operator, and constructing the circuit hierarchy $W_j \rightarrow V_1, V_2 \rightarrow \tilde{V}_1, \tilde{V}_2 \rightarrow V_{\text{adv}}$. Concluding the chapter, just like the previous one, factorizing the Trotter error.

Chapter 6 — Complexity analysis. We derive an oracle free gate complexity bound for both algorithms, establishing Theorems 1 2, and compare the quantum costs with the classical finite-difference baselines trying to identify the dimensional regimes of quantum advantage.

Chapter 7 — Pedagogical Example. Before discussing the numerical results, this chapter attempts to explain measurement in the context of quantum computing. In this specific case, the difference is addressed between directly reading the amplitudes and using two built-in Qiskit functions designed to simulate a real measurement on quantum hardware. The aim is to suggest a correct interpretation of the plotting results in the following chapter.

Chapter 8 — Numerical experiments. We validate the theoretical predictions through Qiskit simulations of both equations, comparing the quantum circuit output against three reference solutions at increasing levels of approximation.

Chapter 9 — Conclusions. We summarise the main contributions, discuss the limitations of the present work, and outline directions for future research.

Appendix. A. We explicitly constructed the algorithm for the Quantum Fourier Transform; a cornerstone of various quantum pipelines, including the one addressed in this work.

Part I

Prerequisites

2 Quantum Computing Background

2.1 Elements of Quantum Computing

This section introduces the quantum-mechanical formalism and the gate-based model, also known as digital, of quantum computation that underpin the circuit constructions presented in Chapters 4 and 5. We assume familiarity with linear algebra over $\mathbb{C}$, but no prior exposure to quantum mechanics is required. Standard references for the material covered here are Nielsen and Chuang [20] and Lin [16].

2.1.a Qubits and Quantum States

Definition 1. *The fundamental unit of quantum information is the* qubit, *whose state space is the two-dimensional complex Hilbert space* $\mathbb{C}^2$.
A pure qubit state is a unit vector

$$|\psi\rangle \;=\; \alpha\,|0\rangle + \beta\,|1\rangle\,, \qquad \alpha,\beta\in\mathbb{C}, \quad |\alpha|^2+|\beta|^2=1, \tag{2.1}$$

where $\{|0\rangle\,,|1\rangle\}$ *is the standard (computational) basis of* $\mathbb{C}^2$.

We use Dirac notation throughout: $|\psi\rangle$ denotes a column vector *ket* while we denote by $\langle\psi| = |\psi\rangle^\dagger$ its conjugate transpose (*bra*), and $\langle\phi|\psi|\phi|\psi\rangle$ is the inner product.

Definition 2. *An* n*-qubit system has state space* $\mathscr{H}_n = (\mathbb{C}^2)^{\otimes n} \cong \mathbb{C}^{2^n}$, *with computational basis* $\{|j\rangle\}_{j=0}^{2^n-1}$, *where* j *is interpreted as its* n*-bit binary representation. (i.e. 5 = 101).*
A general pure state is

$$|\psi\rangle \;=\; \sum_{j=0}^{2^n-1} c_j\,|j\rangle\,, \qquad c_j\in\mathbb{C}, \quad \sum_{j=0}^{2^n-1}|c_j|^2=1.$$

Definition 3. *Given a real vector* $u=(u_0,\ldots,u_{N-1})\in\mathbb{R}^N$ *with* $N=2^n$, *its* amplitude-encoded *quantum state is defined as*

$$|u\rangle \;=\; \frac{1}{\|u\|}\sum_{j=0}^{N-1} u_j\,|j\rangle\,. \tag{2.2}$$

This encoding stores N real numbers in $n = \log_2 N$ qubits. The exponential compression relative to classical memory is one source of quantum advantage for different types of problems including those addressed in this work.

Remark: 1. *It is really important to notice that the amplitude encoding* (2.2) *encodes only the* direction *of u; the norm $\|u\|$ must be tracked separately. Efficient state-preparation circuits are an active research topic and for this reason they will not be discussed in this work.*

As always in mathematics the states $|0\rangle$ and $|1\rangle$ represent just one of many possible choices of basis for a qubit. Other commonly used basis are:

- **Hadamard Basis:**
$$|+\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$$
$$|-\rangle = \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle$$

- **Phase/ Circular Polarization Basis:**
$$|L\rangle = |+i\rangle = \frac{1}{\sqrt{2}}(|0\rangle + i|1\rangle)$$
$$|R\rangle = |-i\rangle = \frac{1}{\sqrt{2}}(|0\rangle - i|1\rangle)$$

More generally, given any basis states $|a\rangle$ and $|b\rangle$ for a qubit, it is possible to express an arbitrary state as a linear combination $\alpha\,|a\rangle + \beta\,|b\rangle$ of those states. Furthermore, provided the states are orthonormal, it is possible to perform a measurement with respect to the $|a\rangle$, $|b\rangle$ basis, giving the result a with probability $|\alpha|^2$ and b with probability $|\beta|^2$. The orthonormality constraint is necessary in order that $|\alpha|^2 + |\beta|^2 = 1$ as we expect for probabilities.
In an analogous way,In an analogous way, it is possible in principle to measure a quantum system of many qubits with respect to an arbitrary orthonormal basis. However, it is relevant to notice that, just because it is possible in principle does not mean that such a measurement can be done easily.

2.1.b Bloch Sphere Representation

One picture useful in thinking about qubits is the following geometric representation. The probability condition in equation (2.1) allows us to define the Bloch Sphere representation of a qubit as

$$|\psi\rangle = e^{i\gamma}\left[\cos\left(\frac{\theta}{2}\right)|0\rangle + e^{i\phi}\sin\left(\frac{\theta}{2}\right)|1\rangle\right] \tag{2.3}$$

Where:

- $\theta \in [0, \pi]$ is the polar angle;
- $\phi \in [0, 2\pi)$ is the azimuthal angle;

- γ is a global phase, often omitted since it cannot be represented on the Bloch sphere directly.

Given the angles to retrieve the cartesian coordinates:

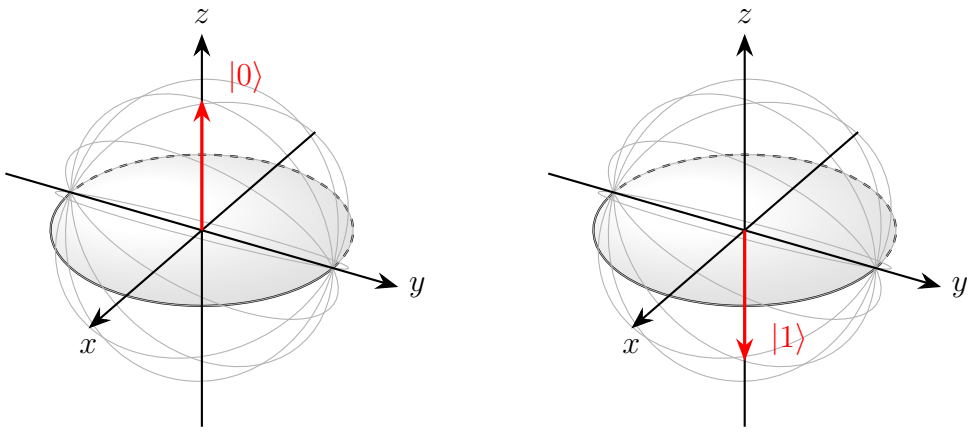
$$x = \sin\theta\cos\phi, \quad y = \sin\theta\sin\phi, \quad z = \cos\theta.$$

Rearranging the Bloch sphere formula, we obtain that θ and ϕ can be expressed as:

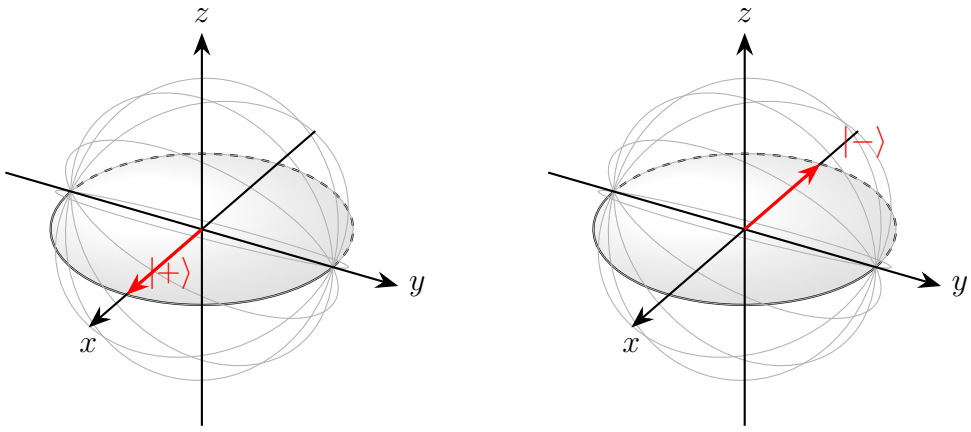
$$\theta = 2\arccos(\alpha_1), \quad \phi = -i\ln\left(\frac{\alpha_2}{\sin\left(\frac{\theta}{2}\right)}\right).$$

As an example if we consider the quantum bases their θ and ϕ values are:

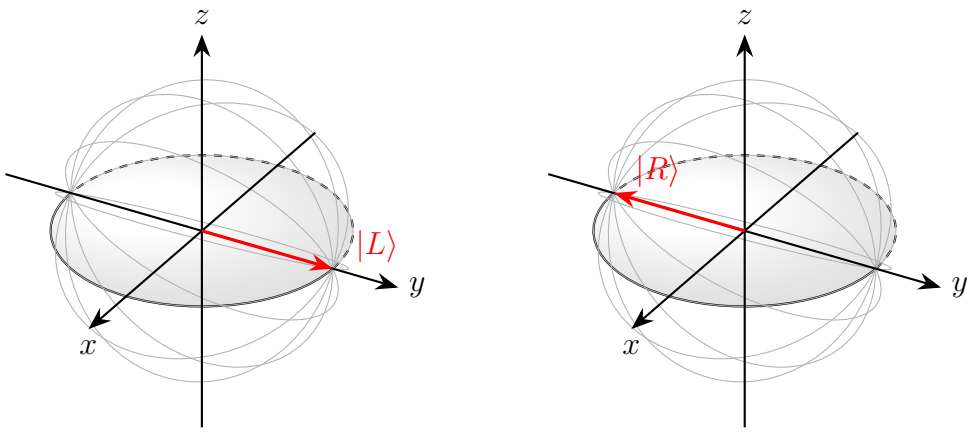
- **Computational Basis:** $|0\rangle \to \theta = 0, \phi = 0, \quad |1\rangle \to \theta = \pi, \phi = 0$



- **Hadamard Basis:** $|+\rangle \to \theta = \frac{\pi}{2}, \phi = 0, \quad |-\rangle \to \theta = \frac{\pi}{2}, \phi = \pi$



- **Phase Basis:** $|L\rangle = |+i\rangle \to \theta = \frac{\pi}{2}, \phi = \frac{\pi}{2}, \quad |R\rangle = |-i\rangle \to \theta = \frac{\pi}{2}, \phi = -\frac{\pi}{2}$



2.1.c Quantum Measurement

When a qubit is measured the quantum state collapses to an eigenstate.The measurement probability depends on squared amplitude, i.e. if we consider

$$P(0) = |\alpha|^2, \quad P(1) = |\beta|^2$$

Post-measurement state:

$$|\psi_{\text{new}}\rangle = \frac{|b\rangle\langle b|\psi\rangle}{\sqrt{P(b)}}$$

As an example consider the state $|\psi\rangle = \frac{1}{\sqrt{3}}|0\rangle + \sqrt{\frac{2}{3}}|1\rangle$:

- Probability of measuring $|0\rangle$: $P(0) = \frac{1}{3}$
- Probability of measuring $|1\rangle$: $P(1) = \frac{2}{3}$

2.2 Quantum Gates and Circuits

There are many physical implementations of qubits, e.g., superconducting qubits, trapped ions, natural atoms, etc. These implementations are a topic for another course. In the following, we assume that we have access to qubits, but it is not of our interest how they are made.

2.2.a Quantum Circuits

Analogous to the way a classical computer is built from an electrical circuit containing wires and logic gates, a quantum computer is built from a quantum circuit containing wires and elementary quantum gates to carry around and manipulate the quantum information. The circuit is to be read from left-to-right. Each line in the circuit represents a wire in the quantum circuit. In the quantum case the wires do not necessarily correspond to a physical wires; they may correspond instead to the passage of time, or perhaps to a physical particle such as a photon moving through space from one location to another. It is conventional to assume, unless otherwise specified, that the state input to the circuit is a computational basis state, usually the state consisting of all $|0\rangle$ s. Even if introducing

quantum circuits starting from classical circuits is helpful from a pedagogical standpoint, they exhibit substantial differences. First of all, the quantum case do not allow 'loops'; we say the circuit is acyclic. Second, classical circuits allow wires to be 'joined' together, an operation known as "FANIN". Observe how such operation is, obviously, not reversible and therefore not unitary, we will see that we only allow unitary operation. Third, the inverse operation "FANOUT", whereby several copies of a bit are produced is also not allowed in quantum circuits. In fact, it turns out that quantum mechanics forbids the copying of a qubit, making the operation impossible.

2.2.b Single-qubit gates

While wires are used to "carry" the information around the logic gates perform manipulations of the information. An operation on a single qubit changes its state from $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ to $|\psi'\rangle = \alpha'|0\rangle + \beta'|1\rangle$ while preserving the norm $|\alpha|^2 + |\beta|^2 = 1 = |\alpha'|^2 + |\beta'|^2$. Such single-qubit operations are what we usually referred to as single gates and can be described by 2×2 unitary matrices. A general such matrix is commonly denoted U. In the following lines we will enlist most common single-qubit gates, both as matrix representation and as gates on circuits. Here we list some of the most common single-qubit gates. First are the Pauli matrices:

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} = \text{—}\boxed{X}\text{—}\,,$$
$$Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} = \text{—}\boxed{Y}\text{—}\,,$$
$$Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} = \text{—}\boxed{Z}\text{—}\,.$$

The Pauli matrices generate rotations around the corresponding axes on the Bloch sphere when exponentiated, e.g.

$$R_x(\theta) = e^{-i\theta X/2} = \cos(\theta/2)I - i\sin(\theta/2)X =$$
$$= \begin{pmatrix} \cos(\theta/2) & -i\sin(\theta/2) \\ -i\sin(\theta/2) & \cos(\theta/2) \end{pmatrix} = \text{—}\boxed{R_X(\theta)}\text{—}\,.$$

It is easy to observe that the Z gate leaves $|0\rangle$ unchanged, and flips the sign of $|1\rangle$ to give $-|1\rangle$ while the X gate is the quantum equivalent of the classical NOT gate.

$$X(\alpha|0\rangle + \beta|1\rangle) = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} \beta \\ \alpha \end{pmatrix} = \beta|0\rangle + \alpha|1\rangle\,. \tag{2.4}$$

We can obtain one of the most famous gates, the Hadamard gate, by adding the gates X and Z.

$$H = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix} = \frac{X+Z}{\sqrt{2}} = \quad -\boxed{H}- \,. \tag{2.5}$$

This gate transforms the qubit state from the $|0\rangle, |1\rangle$ basis to the $|+\rangle, |-\rangle$ basis, and vice versa. Such a gate is particularly relevant since is often used to create superposition states at the beginning of a quantum algorithm.

Two gates that apply other phase factors are often given their own names. One is the T, or $\pi/8$, gate:

$$T = \begin{pmatrix}1 & 0\\ 0 & \exp(i\pi/4)\end{pmatrix} = \exp(i\pi/8)\begin{pmatrix}\exp(-i\pi/8) & 0\\ 0 & \exp(i\pi/8)\end{pmatrix} = \quad -\boxed{T}- \,. \tag{2.6}$$

The other is the phase, or S, or P, gate:

$$S = \begin{pmatrix}1 & 0\\ 0 & i\end{pmatrix} = T^2 = \quad -\boxed{S}- \,. \tag{2.7}$$

2.2.c Multi-qubit gates

To realize useful quantum algorithms, we also need to be able to make two or more qubits interact through multi-qubit gates. To discuss this topic, we first need to establish some notation for states of multiple qubits and operators acting on them. We will not focus on the math behind the tensor product, $\otimes$, but a good understanding of it is required, so we encourage the reader to have a look at it, we suggest [15] (Chapter XVI), before committing to this section.

Briefly, if we have two qubits a and b with states $|\psi_a\rangle = a_0 |0\rangle_a + a_1 |1\rangle_a$ and $|\psi_b\rangle = b_0 |0\rangle_b + b_1 |1\rangle_b$, their total state is

$$|\psi_a\rangle \otimes |\psi_b\rangle = \begin{pmatrix}a_0\\ a_1\end{pmatrix} \otimes \begin{pmatrix}b_0\\ b_1\end{pmatrix} = \begin{pmatrix}a_0\begin{pmatrix}b_0\\ b_1\end{pmatrix}\\ a_1\begin{pmatrix}b_0\\ b_1\end{pmatrix}\end{pmatrix} = \begin{pmatrix}a_0 b_0\\ a_0 b_1\\ a_1 b_0\\ a_1 b_1\end{pmatrix} =$$
$$= a_0 b_0 |00\rangle + a_0 b_1 |01\rangle + a_1 b_0 |10\rangle + a_1 b_1 |11\rangle \,,$$

where $|01\rangle$ denotes $|0\rangle_a \otimes |1\rangle_b$. We can reitarate the same argument for higher dimensional qubits.

Whenever a X gate acts on qubit a (and does nothing, i.e., identity, to qubit b) and another X gate acts on qubit b (and does nothing, i.e., identity, to qubit a) in this state, we can write it as:

$$(X\otimes I_2)(I_2\otimes X)\left(|\psi_a\rangle \otimes |\psi_b\rangle\right) = (X\otimes X)\left(|\psi_a\rangle \otimes |\psi_b\rangle\right) = (X|\psi_a\rangle)\otimes(X|\psi_b\rangle) = X_a X_b |\psi_a\psi_b\rangle \,,$$

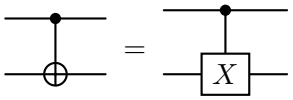
where I_2 is the 2×2 identity matrix. Note that we will mostly omit tensor product

signs and use abbreviated notation like in the rightmost expression in (2.2.c). Also note that entangled states of qubits, like the ones produced by many two-qubit gates, are not separable, i.e., cannot be written as the tensor product of individual single-qubit states (like $|\psi_a\rangle \otimes |\psi_b\rangle$ above).

Remark: 2. *Observe that the two-qubit Hilbert space is spanned by the basis vectors* $|00\rangle$, $|01\rangle$, $|10\rangle$, *and* $|11\rangle$, *in that order. This is the tensor product of the two single-qubit Hilbert spaces. These four states are the computational basis, and this is the basis we will use when writing all the following multi-qubit gates. However, as in the single case, one could also choose another basis for the two-qubit Hilbert space, e.g.,* $|{+}{+}\rangle$, $|{+}{-}\rangle$, $|{-}{+}\rangle$, *and* $|{-}{-}\rangle$.

Now, one way to achieve an interaction between two qubits is to let the state of one qubit control whether a certain single-qubit gate is applied to another qubit. One such gate is the controlled-NOT (CNOT) gate:

$$\text{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} =$$

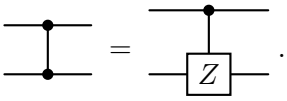


. (2.8)

The action of the CNOT gate in (2.8) is thus to do nothing if the first qubit is in state $|0\rangle$ ($|00\rangle$, $|01\rangle$ changes to $|00\rangle$, $|01\rangle$), and to apply the logic NOT, i.e the X gate, to the second qubit if the first qubit is in state $|1\rangle$ ($|10\rangle$, $|11\rangle$ changes to $|11\rangle$, $|10\rangle$).
Conventionally, in the quantum circuit notation the filled dot in the diagram denotes the *control qubit*; in the literature one can also encounter an empty dot, in that case the operation is performed on the *targets qubits* if the control qubit is in the state $|0\rangle$.

The controlled-Z (CZ) gate can be defined in the same manner:

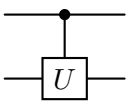
$$\text{CZ} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix} =$$



. (2.9)

Here, both qubits can be seen as controls, since something only happens in the $|11\rangle$ state (it gets a factor -1). More generally, the controlled application of a single-qubit unitary U to the second qubit takes the form

$$\begin{pmatrix} I_2 & 0_2 \\ 0_2 & U \end{pmatrix} =$$



. (2.10)

There are also two-qubit gates that are not controlled operations. For example, the

SWAP gate

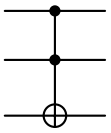
$$\text{SWAP} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \tag{2.11}$$

swaps the states $|01\rangle$, $|10\rangle$ to $|10\rangle$, $|01\rangle$.

It is intuitive to imagine that it is also possible to define gates involving more than two qubits, such as multiple controlled gates. Here we present the most well-known ones in three-qubit case .

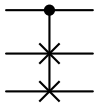
The Toffoli gate is a controlled-controlled-NOT (CCNOT), i.e., the state of the third qubit is flipped if and only if both the first two qubits are in state $|1\rangle$:

$$\text{Toffoli} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} = \tag{2.12}$$



The Fredkin gate is a controlled-SWAP gate (CSWAP), swapping the states of the second and third qubits if and only if the state of the first qubit is $|1\rangle$:

$$\text{Fredkin} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} = \tag{2.13}$$



2.2.d Universal quantum gates

At this point in our dissertation we should address the first elephant in the room: a fundamental tension in quantum computing is the gap between hardware constraints (limited connectivity) and theoretical capabilities (universal gate sets). Indeed, most practical implementations of quantum computing have limited connectivity between qubits, only allowing for pairwise interactions between nearest-neighbor qubits. This forbid direct implementations of multi-qubit gates with three or more qubits. A partial answer to this question is given by the Universal Gate Theorems (like the Solovay-Kitaev theorem) that prove how a small set of 1-qubit and 2-qubit gates (like CNOT, H, T, and S), known as universal quantum gates, can approximate any multi-qubit unitary operation to arbitrary

precision. A proof of the Solovay-Kitaev theorem can be found [20].
While universality solves the “can we decompose a 3-qubit gate into 2-qubit gates?" problem, it assumes that one can apply those 2-qubit gates to any two qubits in the system.

In real-world architectures (like superconducting qubits or quantum dots), qubits are arranged on a fixed grid (e.g., a heavy-hex or square lattice) and they can only directly interact with their nearest neighbors. When one needs to apply a controlled gate (like CNOT) between two qubits that are not adjacent, he faces a connectivity constraint, and so universality alone do not solve the problem. To fill this gap Quantum compilers use Quantum Circuit Routing (or Mapping): suppose that the Qubit A needs to interact with Qubit C, but they are separated by Qubit B, the compiler inserts a SWAP gate between A and B. Now, the state of Qubit A is sitting physically next to C. After the interaction is complete, more SWAP gates are used to move the information back.
We will not address this issue further, as the central algorithm discussed in this thesis is evidently designed for a Fault-Tolerant Quantum Computing (FTQC). However, at this introductory stage, we deemed appropriate to highlight to the reader that such problems currently exist and must be taken into consideration whenever a quantum algorithm is designed.
There is, indeed, a huge difference between the “Ideal Circuit" drawn on paper and the “Physical Circuit" the machine actually runs. Ideally, one uses logical gates as building blocks drawing a CNOT gate between qubits that live far away, and quantum programming framework (like Qiskit or Cirq) will accept it without throwing an error. However, before the quantum computer runs the code, a software program called transpiler takes the logical circuit and rewrites it to match the actual hardware. If your logical circuit uses a gate that the hardware does not natively support, or connects two qubits that are not neighbors, the transpiler fixes it by adding extra gates; in such way the computational cost skyrockets behind your back.

2.2.e Gate count and circuit complexity

The gate complexity of a quantum algorithm is measured by the total number of elementary quantum gates required to implement it. We adopt the standard convention of counting:

- Single-qubit gates: H, $P(\lambda)$, $R_Z(\theta)$, X, Y, Z, and their controlled versions with one control qubit;
- Two-qubit (non-local) gates: CNOT gates.

Multi-controlled R_Z or R_X gates with $(k-1)$ control qubits can be decomposed into at most $16k - 40$ CNOT gates and a comparable number of single-qubit gates using the method of Vale et al. [22]. This decomposition is used systematically in the complexity analysis of Chapter 6.

2.3 Hamiltonian Simulation

Definition 4 (Hamiltonian simulation problem)**.** *Given a Hermitian matrix $H \in \mathbb{C}^{N\times N}$, a total evolution time $T > 0$, and a target precision $\varepsilon > 0$, the* Hamiltonian simulation problem *asks for a quantum circuit V such that*

$$\left\| V - e^{iHT} \right\| \;\leq\; \varepsilon,$$

where the norm is the operator (spectral) norm.

The importance of Hamiltonian simulation stems from the Schrödinger equation of quantum mechanics, $\partial_t |\psi\rangle = -iH|\psi\rangle$, whose solution is $|\psi(T)\rangle = e^{-iHT}|\psi(0)\rangle$. For PDEs, the Schrödingersation technique (Chapter 3) reduces general linear evolution equations to Hamiltonian simulation problems, making Definition 4 the central computational task of this thesis.

Product formula methods. When H decomposes as $H = \sum_{\ell=1}^{L} H_\ell$, the simplest approximation of e^{iHt} is the *first-order Lie–Trotter–Suzuki* (LTS) product formula,

$$e^{iHt} \;\approx\; \prod_{\ell=1}^{L} e^{iH_\ell t} \;=:\; V_{\mathrm{LTS}}^{(1)}(t), \tag{2.14}$$

with error

$$\left\| e^{iHt} - V_{\mathrm{LTS}}^{(1)}(t) \right\| \;\leq\; \frac{t^2}{2} \sum_{\ell<\ell'} \|[H_\ell, H_{\ell'}]\| \;=\; \mathcal{O}(t^2), \tag{2.15}$$

as shown by Childs et al. [6] (V.A. Proposition 9) using commutator scaling. To simulate e^{iHT} within precision ε, one divides the total time into $r = T/\tau$ steps and applies (2.14) with step size τ:

$$\left\| e^{iHT} - \left(V_{\mathrm{LTS}}^{(1)}(\tau)\right)^r \right\| \;\leq\; r \cdot \left\| e^{iH\tau} - V_{\mathrm{LTS}}^{(1)}(\tau) \right\| \;=\; \mathcal{O}\!\left(\frac{T^2}{r}\right).$$

Setting $r = \mathcal{O}(T^2/\varepsilon)$ thus suffices to achieve precision ε, at a gate cost that is linear in r. Higher-order Suzuki formulas reduce the step count at the cost of a larger number of exponential factors per step [3].

Remark: 3. *The commutator bound (2.15) is tighter than the naive triangle-inequality estimate $\mathcal{O}(t^2 L^2 \max_\ell \|H_\ell\|^2)$ whenever the summands H_ℓ nearly commute, which is precisely the case for the spatially local operators arising from finite difference discretizations of PDEs. The explicit commutator calculations required for our error bounds are carried out in Appendix B of the article [14].*

State-of-the-art algorithms Beyond product formulas, several approaches achieve optimal or near-optimal dependence on the simulation parameters. Berry et al. [2] combined quantum walk techniques with a fractional-query approach to obtain an algorithm with

gate complexity

$$\mathcal{O}\left(\frac{\tau}{\log(\tau/\varepsilon)/\log\log(\tau/\varepsilon)}\right), \qquad \tau := s\|H\|_{\max}T,$$

where we denote by s the sparsity of H and $\|H\|_{\max}$ the largest entry in absolute value. This is near-linear in τ and achieves exponential improvement over ε compared to product formulas. Further refinements include Taylor-series methods, qubitisation, and quantum signal processing. For our purposes the first-order LTS formula suffices to demonstrate quantum advantage, indeed we will see how the dominant cost comes from the spatial dimension d rather than from the simulation precision.

2.4 Quantum Fourier Transform

Definition 5 (Quantum Fourier Transform)**.** *In the Schrödingersation pipeline (Figure 3.2) a central role is played by the Quantum Fourier Transform (QFT). The n-qubit Quantum Fourier Transform (QFT) is the unitary operator $F_n : \mathbb{C}^{2^n} \to \mathbb{C}^{2^n}$ defined by its action on computational basis states:*

$$F_n\,|j\rangle \;=\; \frac{1}{2^{n/2}}\sum_{k=0}^{2^n-1}\omega_n^{jk}\,|k\rangle\,, \qquad \omega_n = e^{2\pi i/2^n}, \quad j = 0,\ldots,2^n-1. \tag{2.16}$$

The QFT is the discrete Fourier transform (DFT) of the amplitude vector, and can be implemented exactly using the $\mathcal{O}(n^2)$ single-qubit and CNOT gates [20], compared to $\mathcal{O}(N\log N) = \mathcal{O}(2^n n)$ for the classical FFT. Due to the relevance, not only in our algorithm but also as backbone of algorithms such as Shor's Algorithm, the full construction is explained in Appendix A

2.5 Quantum Measurement and Amplitude Estimation

Projective measurements A projective measurement with respect to an orthonormal basis $\{|e_k\rangle\}$ yields outcome k with probability $p_k = |\langle e_k|\,\psi\rangle|^2$ (Born rule), collapsing the state to $|e_k\rangle$.

Observable estimation For an observable (Hermitian operator) O, the expected value $\langle O\rangle = \langle\psi|\,O\,|\psi\rangle$ can be estimated to within additive error δ using $\mathcal{O}(1/\delta^2)$ repeated circuit executions (*shots*) and classical averaging.

Post-selection overhead After the $M_{\geq 0}$ measurement, the desired state is obtained with probability

$$\Pr(\text{success}) \;=\; \frac{\|u(T)\|^2\,\|p_{\geq 0}\|^2}{\|u(0)\|^2\,\|p\|^2} \;\approx\; \frac{\|u(T)\|^2}{\|u(0)\|^2}, \tag{2.17}$$

so $\mathcal{O}(\|u(0)\|^2/\|u(T)\|^2)$ repetitions are needed to obtain a single successful sample.

Part II

Resuts

3 Schrödingerization: Theory and Discretization

Quantum simulation represents one of the most natural applications of quantum devices, exploiting their physical evolution to reproduce outputs governed by Schrödinger's equations. An intuitive broader question concerns the ability of quantum devices for simulating more general dynamical laws, expressed both as ordinary or partial differential equations (ODEs or PDEs). Many such equations, characterized by high dimensionality or multiple temporal and spatial scales, have prohibitive computational costs on classical hardware, making the development of quantum algorithms for their solution highly desirable.

Existing approaches carry notable limitations. Quantum algorithms for linear algebraic systems can yield significant speedup in solving discretized PDEs, yet quantum simulation in this context serves only as an algorithmic primitive rather than a continuous-time solver. Qubitization and block encoding methods can approximate arbitrary non-unitary operators, but require complex polynomial constructions ill-suited to continuous-variable architectures.

To address these limitations Jin, Liu, and Yu [12, 13] have proposed a novel paradigm based on the warped phase transformation, which maps any linear PDE or ODE (including dissipative, time-irreversible, and non-autonomous systems) to a Schrödinger equation in real time, via an embedding in one higher dimension followed by a Fourier transform. This approach, which they term Schrödingerization, enables the preparation of solutions at any desired time t as a quantum state. Beyond classical PDEs and ODEs, they claim that the method also applies to quantum ground and thermal state preparation, semiclassical simulations, discrete dynamical systems, and linear algebra, while naturally supporting qubits, continuous-variable systems, hybrid architectures, and qudits a versatility not shared by existing methods.

This chapter develops the method in full generality before implementing it for concrete PDEs treated in Chapters 4 and 5. Section 3.1 Section 3.2 establishes the continuous-level theory, including the precise sense in which the transformation is invertible and the regularity conditions that govern the approximation quality. Section 3.2.b carries out the discretizations of the auxiliary p-variable, deriving the spectral error estimates that feed into the complexity analysis of Chapter 6. Section 3.2.c gives a coordinate-free description of the full quantum pipeline: state preparation, the QFT over the ancilla register, iterated

Hamiltonian simulation, the inverse QFT, and the post-selection measurement.

3.1 Encoding of Finite Difference Operators as Quantum Operators

We develop the technical core that links classical PDE numerics to quantum circuits, we will show how the finite difference operators can be expressed in terms of elementary quantum gates acting on an n_x-qubit register. Throughout this section we assume a uniform one-dimensional grid of $N_x = 2^{n_x}$ points (generalization to d dimensions are easily obtained by tensor products, as shown in Section 4.3).

3.1.a Shift Operators and their decomposition

Conventionally, we denote the 2×2 matrices generated by the standard basis qubit as

$$\sigma_{01} := |0\rangle\langle 1|, \quad \sigma_{10} := |1\rangle\langle 0|, \quad \sigma_{00} := |0\rangle\langle 0|, \quad \sigma_{11} := |1\rangle\langle 1|. \tag{3.1}$$

we also define

$$s_j^- := I^{\otimes(n_x-j)} \otimes \sigma_{01} \otimes \sigma_{10}^{\otimes(j-1)}, \tag{3.2}$$

$$s_j^+ := I^{\otimes(n_x-j)} \otimes \sigma_{10} \otimes \sigma_{01}^{\otimes(j-1)}. \tag{3.3}$$

Definition 6. *Consider the Hilbert space $\mathscr{H}_{n_x} = (\mathbb{C}^2)^{\otimes n_x}$ and let its computational basis states be indexed by integers $j \in \{0, 1, \ldots, n_x - 1\}$, denoted as $|j\rangle$. The general discrete shift operators, left-shift and right-shift, can be defined respectively as:*

$$S^- |j\rangle := |j-1\rangle,$$
$$S^+ |j\rangle := |j+1\rangle;$$

They can be implemented as:

$$S^- := \sum_{j=1}^{2^{n_x}-1} |j-1\rangle\langle j| = \sum_{j=1}^{n_x} I^{\otimes(n_x-j)} \otimes \sigma_{01} \otimes \sigma_{10}^{\otimes(j-1)} = \sum_{j=1}^{n_x} s_j^-, \tag{3.4}$$

$$S^+ := \sum_{j=1}^{2^{n_x}-1} |j\rangle\langle j-1| = \sum_{j=1}^{n_x} I^{\otimes(n_x-j)} \otimes \sigma_{10} \otimes \sigma_{01}^{\otimes(j-1)} = \sum_{j=1}^{n_x} s_j^+. \tag{3.5}$$

Recall that the forward and backward finite difference operators are defined as follows:

$$(D^+\mathbf{u})_j = \frac{u_{j+1} - u_j}{h}, \qquad (D^-\mathbf{u})_j = \frac{u_j - u_{j-1}}{h}, \tag{3.6}$$

for $j = 0, 1, \ldots, N_x - 1$.

Accordingly, the forward (D_P^+), backward (D_P^-), and central ($D_P^\pm$) finite difference operators with respect to the first-order derivative, as well as the central finite difference oper-

ator (D_P^{Δ}) with respect to the second-order derivative, combined with periodic boundary conditions, can be expressed as:

$$\begin{aligned} D_P^+ &= \frac{S^- - I^{\otimes n_x} + \sigma_{10}^{\otimes n_x}}{h}, \\ D_P^- &= \frac{I^{\otimes n_x} - S^+ - \sigma_{01}^{\otimes n_x}}{h}, \\ D_P^{\pm} &= \frac{S^- - S^+ - \sigma_{01}^{\otimes n_x} + \sigma_{10}^{\otimes n_x}}{2h}, \\ D_P^{\Delta} &= \frac{S^- + S^+ - 2I^{\otimes n_x} + \sigma_{01}^{\otimes n_x} + \sigma_{10}^{\otimes n_x}}{h^2}. \end{aligned} \tag{3.7}$$

Where the correction terms σ are needed in order to cancel the erroneous wrap-around and replace it with the correct periodic contribution.
Similarly, the finite difference operator incorporating Dirichlet boundary conditions on either the left or right boundary can be expressed as:

$$\begin{aligned} D_D^+ &= \frac{S^- - I^{\otimes n_x}}{h} \qquad \text{(right side)}, \\ D_D^- &= \frac{I^{\otimes n_x} - S^+}{h} \qquad \text{(left side)}. \end{aligned} \tag{3.8}$$

Considering Dirichlet boundary conditions on both sides, i.e., $u_0 = u_{N_x} = 0$, it suffices to solve for the numerical solution $\mathbf{u} := [u_1, \cdots, u_{N_x-1}]$. In this case, we set $N_x = 2^{n_x} + 1$ and define $|\mathbf{u}\rangle = \mathbf{u}/\|\mathbf{u}\|$, with $\mathbf{u} := \sum_{j=0}^{2^{n_x}-1} u_{j+1}|j\rangle$. Consequently, the finite difference operators with Dirichlet boundary conditions for both sides are:

$$\begin{aligned} D_D^+ &= \frac{S^- - I^{\otimes n_x}}{h}, \\ D_D^- &= \frac{I^{\otimes n_x} - S^+}{h}, \\ D_D^{\pm} &= \frac{S^- - S^+}{2h}, \\ D_D^{\Delta} &= \frac{S^- + S^+ - 2I^{\otimes n_x}}{h^2}. \end{aligned} \tag{3.9}$$

3.1.b Key Lemmas for Circuit Implementation

The following three lemmas, due to Sato et al. [21] and extended in [14], provide the building blocks for all circuit constructions in this thesis.

Lemma 1. *Let $|a\rangle, |b\rangle \in \mathbb{C}^{2^n}$ be orthogonal unit vectors. The operator $S = |a\rangle\langle b| - |b\rangle\langle a|$ and can be written as*

$$S = \left|a'\right\rangle\!\left\langle a'\right| - \left|b'\right\rangle\!\left\langle b'\right|, \qquad \left|a'\right\rangle = \frac{|a\rangle + |b\rangle}{\sqrt{2}}, \quad \left|b'\right\rangle = \frac{|a\rangle - |b\rangle}{\sqrt{2}}. \tag{3.10}$$

Moreover, there exists a unitary B such that $B|0\rangle|1\rangle^{\otimes(n-1)} = |a'\rangle$, $B|1\rangle^{\otimes n} = |b'\rangle$, and

$$e^{iSt} = B \cdot \mathrm{CRZ}^n_{1,\dots,n-1}(-2t) \cdot B^\dagger. \tag{3.11}$$

Proof. Let $S = |a\rangle\langle b| + |b\rangle\langle a|$ s.t. $\langle a|b|a|b\rangle = 0$. Define

$$|a'\rangle := \frac{|a\rangle + |b\rangle}{\sqrt{2}}, \quad |b'\rangle := \frac{|a\rangle - |b\rangle}{\sqrt{2}}.$$

Consequently we have

$$|a\rangle := \frac{|a'\rangle + |b'\rangle}{\sqrt{2}}, \quad |b'\rangle := \frac{|a'\rangle - |b'\rangle}{\sqrt{2}}.$$

Therefore the following chain of equality holds:

$$\begin{aligned} S &= |a\rangle\langle b| + |b\rangle\langle a| && (3.12)\\ &= \frac{(|a'\rangle + |b'\rangle)(\langle a'| - \langle b'|)}{2} + \frac{(|a'\rangle - |b'\rangle)(\langle a'| + \langle b'|)}{2} && (3.13)\\ &= \frac{2|a'\rangle\langle a'| - 2|b'\rangle\langle b'|}{2} = |a'\rangle\langle a'| - |b'\rangle\langle b'| && (3.14) \end{aligned}$$

This conclude the first part of the proof.
Let B be the unitary matrix s.t. $B|0\rangle|1\rangle^{\otimes(n-1)} = |a'\rangle$ and $B|1\rangle^{\otimes n} = |b'\rangle$ such matrix exists since $|a'\rangle$, $|b'\rangle$ are orthonormal we simply need to complete the orthonormal basis and construct B having these vectors as columns ensuring that the required position of $|a'\rangle$ and $|b'\rangle$ are satisfied

$$B = \begin{pmatrix} | & | & \cdots & | & \cdots & | \\ \vdots & \vdots & & |a'\rangle & & |b'\rangle \\ | & | & & | & \cdots & | \\ & & & \underbrace{}_{011\dots1} & & \underbrace{}_{111\dots1} \end{pmatrix}$$

Recall that the Pauli operator $Z := \sigma_{00} - \sigma_{11} = |0\rangle\langle 0| - |1\rangle\langle 1|$, multiplying it we obtain

$$Z \otimes |1\rangle\langle 1|^{\otimes n-1} = |0\rangle\langle 0| \otimes |1\rangle\langle 1|^{\otimes n-1} - |1\rangle\langle 1|^{\otimes n}$$

If we now multiply such quantity by B and $B^\dagger$ we get

$$B(Z \otimes |1\rangle\langle 1|^{\otimes n-1})B^\dagger = |a'\rangle\langle a'| - |b'\rangle\langle b'| = S. \tag{3.15}$$

Considering the exponential of (3.15)

$$e^{iSt} = e^{B(Z\otimes|1\rangle\langle 1|^{\otimes n-1})B^\dagger} = Be^{(Z\otimes|1\rangle\langle 1|^{\otimes n-1})}B^\dagger = B(CRZ_n^{1\dots n-1}(-2t))B^\dagger. \tag{3.16}$$

Where the last equality holds since $e^{iZt} = RZ(-2t)$ and we are applying such operation only if the last $n-1$ qubits are equal to $|1\rangle$, i.e. we have a controlled gate. □

Lemma 2 (Circuit for $W_j(\gamma\tau,\lambda)$, [21, Lemma 2]). *For $j \in \{1,\dots,n_x\}$, $\gamma,\tau,\lambda \in \mathbb{R}$, define*

$$W_j(\gamma\tau,\lambda) \;:=\; B_j(\lambda)\,\mathrm{CRZ}^{j}_{1,\dots,j-1}(-2\gamma\tau)\,B_j(\lambda)^{\dagger}, \tag{3.17}$$

where

$$B_j(\lambda) \;:=\; \left(\prod_{m=1}^{j-1}\mathrm{CNOT}^{j}_{m}\right) P_j(-\lambda)\,H_j, \tag{3.18}$$

with H_j the Hadamard gate on qubit j, $P_j(\lambda)$ the phase gate on qubit j acting on this qubit as

$$P_j(\lambda) := \begin{pmatrix} 1 & 0 \\ 0 & e^{i\lambda} \end{pmatrix}, \tag{3.19}$$

and CNOT^{j}_{m} the CNOT gate with control qubit j and target qubit m. Then the time-evolution operator can be implemented explicitly by

$$\exp\!\big(i\gamma\tau\,(e^{i\lambda}s_j^{-} + e^{-i\lambda}s_j^{+})\big) \;=\; I^{\otimes(n_x-j)} \otimes W_j(\gamma\tau,\lambda). \tag{3.20}$$

Proof. Recall the definitions of s_j^- (3.2) and s_j^+ (3.3), direct computation using σ_{01} and σ_{10} shows that

$$e^{i\lambda}s_j^{-} + e^{-i\lambda}s_j^{+} \;=\; I^{\otimes(n_x-j)} \otimes \big(e^{i\lambda}\,|a_j\rangle\langle b_j| + e^{-i\lambda}\,|b_j\rangle\langle a_j|\big),$$

where $|a_j\rangle = |0\rangle\,|1\rangle^{\otimes(j-1)}$ and $|b_j\rangle = |1\rangle\,|0\rangle^{\otimes(j-1)}$. Applying Lemma 1 one can define the unitary matrix $B_j(\lambda)$ s.t.

$$B_j(\lambda)\,|0\rangle\,|1\rangle^{\otimes(j-1)} = \frac{|a\rangle + e^{-i\lambda}|b\rangle}{\sqrt{2}}, \qquad B_j(\lambda)\,|1\rangle\,|1\rangle^{\otimes(j-1)} = \frac{|a\rangle - e^{-i\lambda}|b\rangle}{\sqrt{2}}.$$

We observe that $B_j(\lambda)$ can be constructed as is (3.18), indeed let $|x\rangle$ with $x \in \{0,1\}$, the action of H on $|x\rangle$ is defined as:

$$H\,|x\rangle = \frac{|0\rangle + (-1)^x\,|1\rangle}{\sqrt{2}} \tag{3.21}$$

applying H_j to our initial state yields:

$$\begin{aligned} H_j\left(|x\rangle_j \otimes |1\rangle^{\otimes(j-1)}\right) &= \left(\frac{|0\rangle_j + (-1)^x\,|1\rangle_j}{\sqrt{2}}\right) \otimes |1\rangle^{\otimes(j-1)} \\ &= \frac{1}{\sqrt{2}}\left(|0\rangle_j \otimes |1\rangle^{\otimes(j-1)} + (-1)^x\,|1\rangle_j \otimes |1\rangle^{\otimes(j-1)}\right) \end{aligned} \tag{3.22}$$

The phase gate $P_j(\lambda)$ adds a phase of $e^{i\lambda}$ to the $|1\rangle$ state. Therefore in our case we get

$$\begin{aligned} &P_j(-\lambda)\left[\frac{1}{\sqrt{2}}\left(|0\rangle_j \otimes |1\rangle^{\otimes(j-1)} + (-1)^x\,|1\rangle_j \otimes |1\rangle^{\otimes(j-1)}\right)\right] \\ &= \frac{1}{\sqrt{2}}\left(|0\rangle_j \otimes |1\rangle^{\otimes(j-1)} + (-1)^x e^{-i\lambda}\,|1\rangle_j \otimes |1\rangle^{\otimes(j-1)}\right) \end{aligned} \tag{3.23}$$

Finally, the operator $\prod_{m=1}^{j-1}\mathrm{CNOT}^{j}_{m}$ consists of CNOT gates where the j-th qubit is the

control and the m-th qubits (1 through $j-1$) are the targets.

- If we consider the first term the control qubit is $|0\rangle_j$. The CNOT gates do not alter the target qubits, meaning $|1\rangle^{\otimes(j-1)}$ remains unchanged:

$$|0\rangle_j \otimes |1\rangle^{\otimes(j-1)} \xrightarrow{\text{CNOT}} |0\rangle_j \otimes |1\rangle^{\otimes(j-1)} = |a\rangle \tag{3.24}$$

- On the second term the control qubit is $|1\rangle_j$. The CNOT gates flip all target qubits from $|1\rangle$ to $|0\rangle$:

$$|1\rangle_j \otimes |1\rangle^{\otimes(j-1)} \xrightarrow{\text{CNOT}} |1\rangle_j \otimes |0\rangle^{\otimes(j-1)} = |b\rangle \tag{3.25}$$

Applying these synchronous transformations to the complete superposition yields:

$$B_j(\lambda)\,|x\rangle_j\,|1\rangle^{\otimes(j-1)} = \frac{1}{\sqrt{2}}\left(|a\rangle + (-1)^x e^{-i\lambda}\,|b\rangle\right) \tag{3.26}$$

We will need this construction of B later while implementing the code for the whole algorithm. Given B Lemma 1 allows us to formulate the time evolution operator as:

$$\begin{aligned}\exp\left(i\gamma\tau(e^{i\lambda}s_j^- + e^{-i\lambda}s_j^+)\right) &= I^{\otimes(n_x-j)} \otimes B_j(\lambda)\,\mathrm{CRZ}_j^{1,\dots,j-1}(-2\gamma\tau)\,B_j(\lambda)^\dagger \\ &=: I^{\otimes(n_x-j)} \otimes W_j(\gamma\tau,\lambda),\end{aligned} \tag{3.27}$$

□

The quantum circuit for $W_j(\gamma\tau,\lambda)$ is displayed in Figure 3.1 on n_x qubits.

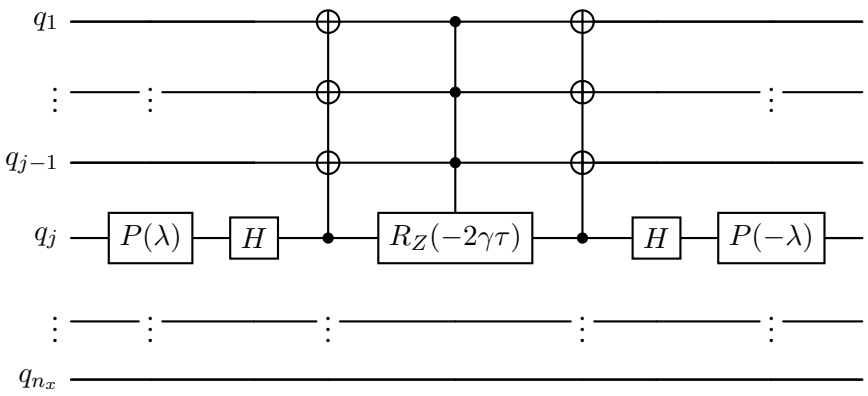


Figure 3.1: Quantum circuit implementing $I^{\otimes(n_x-j)} \otimes W_j(\gamma\tau,\lambda)$

For $j=1$ no control qubits are present and the gate reduces to a plain $R_Z(-2\gamma\tau)$.

Lemma 3 (Controlled-power implementation). *Let $Q_0^k(\tau)$ be a time-evolution operator parametrized by τ, acting on a register of n_p qubits encoding the integer $k \in \{0,\dots,N_p-1\}$, where $N_p = 2^{n_p}$. Using the binary representation $k = (k_{n_p-1}\cdots k_0) = \sum_{m=0}^{n_p-1} k_m 2^m$, the operator $Q_0^k(\tau)$ factorizes as*

$$Q_0^k(\tau) = \prod_{m=0}^{n_p-1} Q_0^{k_m 2^m}(\tau), \tag{3.28}$$

so that the global operator $Q = \sum_{k=0}^{N_p-1} Q_0^k(\tau) \otimes |k\rangle\langle k|$ can be written as a product of n_p elementary controlled gates, yielding an exponential reduction in circuit complexity.

Proof. We begin by expanding Q in the computational basis of the n_p-qubit register:

$$Q = \sum_{k=0}^{N_p-1} Q_0^k(\tau) \otimes |k\rangle\langle k|. \tag{3.29}$$

Substituting the binary decomposition $k = \sum_{m=0}^{n_p-1} k_m 2^m$ and the corresponding factorization $Q_0^k(\tau) = \prod_{m=0}^{n_p-1} Q_0^{k_m 2^m}(\tau)$, we rewrite the sum over k as a nested sum over the individual binary digits $k_{n_p-1}, \ldots, k_0 \in \{0, 1\}$:

$$Q = \sum_{k_{n_p-1},\ldots,k_0} \prod_{m=0}^{n_p-1} Q_0^{k_m 2^m}(\tau) \otimes \Big(|k_{n_p-1}\rangle\langle k_{n_p-1}|\Big) \otimes \cdots \otimes \Big(|k_0\rangle\langle k_0|\Big). \tag{3.30}$$

Since each factor $Q_0^{k_m 2^m}(\tau) \otimes |k_m\rangle\langle k_m|$ depends only on the m-th binary digit k_m, the nested sum factorises into an ordered product over m. Summing over $k_m \in \{0, 1\}$ for each m independently, we obtain

$$Q = \prod_{m=0}^{n_p-1}{}' \left(Q_0^{k_m 2^m}(\tau) \otimes \sum_{k_m} |k_m\rangle\langle k_m| \right), \tag{3.31}$$

where $\prod'$ denotes the fact that the product is a standard matrix product on the first register (of n_x qubits) and a tensor product on the second register (of n_p qubits). Finally, recognizing that $\sum_{k_m} |k_m\rangle\langle k_m| = |1\rangle\langle 1|$ acts as a projector onto the $k_m = 1$ subspace, while the complementary projector $|0\rangle\langle 0|$ contributes the identity $I^{\otimes n_x}$ on the first register, each factor in the product takes the form

$$Q_0^{2^m}(\tau) \otimes |1\rangle\langle 1| + I^{\otimes n_x} \otimes |0\rangle\langle 0|, \tag{3.32}$$

which is precisely the structure of a controlled-$Q_0^{2^m}(\tau)$ gate. This completes the proof that Q decomposes into a product of n_p controlled gates, each acting on a single qubit of the index register, thereby reducing the gate count from $\mathcal{O}(N_p)$ to $\mathcal{O}(n_p) = \mathcal{O}(\log N_p)$. □

Remark: 4. *In the most favourable case, when $Q_0^k(\tau) = Q_0(k\tau)$, each factor in (3.31) is a controlled-Q_0 gate, and the total cost scales as $n_p \cdot \mathrm{cost}(c\text{-}Q_0)$, exponentially cheaper than implementing Q directly. In our heat-equation and advection-equation circuits, the identity $\tilde{V}_0^k(\tau) \neq \tilde{V}_0(k\tau)$ prevents this simplification, but the decomposition (3.31) still gives a quadratic speed-up (in N_p) over a naive direct implementation.*

3.1.c Gate Count for the Building Blocks

We collect here the CNOT counts for the basic building blocks, which are used in the complexity analysis of Chapter 6.

Proposition 1 (CNOT count for W_j)**.** *For $j \geq 2$, the gate $W_j(\gamma\tau, \lambda)$ is implemented using*

- 2 *Hadamard gates*, 2 *Phase gates (if $\lambda \neq 0$);*
- 1 *multi-controlled R_Z gate with $j-1$ control qubits, decomposable into at most $16j-40$ CNOT gates for $j \geq 3$ (and 1 CNOT for $j = 2$);*
- $2(j-1)$ *CNOT gates from the* CNOT_m^j *factors in $B_j(\lambda)$ and $B_j(\lambda)^\dagger$.*

The total CNOT count for W_j is therefore at most $16j - 40 + 2(j-1) = 18j - 42$ for $j \geq 3$, and $2(j-1)+1$ for $j \leq 2$.

Proof. The bound on the multi-controlled R_Z gate follows from the decomposition technique of Vale et al. [22]. The remaining counts are immediate from the circuit of Figure 3.1. □

3.2 The Warped Phase Transformation

Let $\Omega \subset \mathbb{R}^d$ be a bounded domain and let $A : \mathcal{D}(A) \subset L^2(\Omega) \to L^2(\Omega)$ be a (possibly unbounded) linear operator generating a C_0-semigroup $\{e^{At}\}_{t\geq 0}$. The class of operators admissible to Schrödingerisation contains all A for which the decomposition $A = H_1 + iH_2$ into its self-adjoint real and imaginary parts is well-defined, this includes the heat operator $a\Delta$ and the upwind-discretized advection operator.

3.2.a Continuous Formulation

Let $u : [0, T] \to L^2(\Omega)$ be the solution of

$$\partial_t u(t) = Au(t), \qquad u(0) = u_0 \in L^2(\Omega). \tag{3.33}$$

Introduce the auxiliary variable $p \in \mathbb{R}$ and define, for $p \geq 0$, the *warped field*

$$v(t, x, p) := e^{-p}\, u(t, x), \tag{3.34}$$

so that v satisfies

$$\partial_t v(t, x, p) = A_x\, v(t, x, p) = -A_x\, \partial_p\, v(t, x, p), \qquad p > 0, \tag{3.35}$$

where the second equality holds due to the identity $\partial_p(e^{-p}(Au)) = -A_x(e^{-p}u)$ together with $A_x e^{-p} = e^{-p}A_x$ when A_x acts only on x. The equation (3.35) can be extended to all $p \in \mathbb{R}$ by the antisymmetric initial datum

$$v(0, x, p) = e^{-|p|}\, u_0(x), \tag{3.36}$$

which ensures that the full problem on $\mathbb{R}$ is well-posed without additional boundary conditions at $p = 0$.

One can always decompose A into an Hermitian and a skew-Hermitian terms:

$$A = H_1 + iH_2, \quad \text{where} \quad H_1 = \frac{A + A^\dagger}{2} = H_1^\dagger, \quad H_2 = \frac{A - A^\dagger}{2i} = H_2^\dagger. \tag{3.37}$$

A direct calculation shows that the extended field satisfies

$$\partial_t v = -H_1 \partial_p v + iH_2 v. \tag{3.38}$$

Applying the Fourier transform $\mathcal{F}_p$ in the p-variable, $\widehat{v}(t,x,\eta) := \mathcal{F}_p[v(t,x,\cdot)](\eta)$, converts (3.38) into

$$\partial_t \widehat{v}(t,x,\eta) = i\big(\eta H_1 + H_2\big)\widehat{v}(t,x,\eta) =: i\,H(\eta)\,\widehat{v}(t,x,\eta), \tag{3.39}$$

where $H(\eta) := \eta H_1 + H_2$ is Hermitian for every $\eta \in \mathbb{R}$, since both H_1 and H_2 are. Equation (3.39) is a Schrödinger equation parametrised by η, with time evolution operator $e^{iH(\eta)t}$ for each fixed η.

Remark: 5. *When $A = a\Delta$ is the heat operator ($H_2 = 0$), the Hamiltonian reduces to $H(\eta) = \eta a\Delta$, which is self-adjoint under Dirichlet boundary conditions and gives rise to the Hamiltonians studied in Chapter 4. When A arises from an upwind discretisation of the advection operator (Section 5.1.b), both H_1 and H_2 are non-trivial, leading to a more complex circuit structure discussed in Chapter 5.*

In order to get back the solution u a few more steps are required. Applying $\mathcal{F}_p^{-1}$ to the solution $\widehat{v}(t,x,\eta)$ of (3.39) and projecting onto $\{p \geq 0\}$ yields

$$u(t,x) = e^{p_0} \cdot \frac{v(t,x,p_0)}{\|v(0,x,\cdot)|_{p\geq 0}\|_{L^2}} \cdot \|u(0,x)\|_{L^2}, \qquad p_0 \geq 0. \tag{3.40}$$

In the quantum-computational setting, the post-selection measurement $M_{\geq 0}$ realises this projection, at a sampling cost inversely proportional to the probability (2.17).

3.2.b Discretisation of the p-Variable

We discretise the p-axis on the interval $[-\pi R, \pi R]$ with $N_p = 2^{n_p}$ uniformly spaced points

$$p_k = -\pi R + k\,\Delta p, \qquad \Delta p = \frac{2\pi R}{N_p}, \qquad k = 0, 1, \ldots, N_p - 1, \tag{3.41}$$

and Fourier wavenumbers

$$\eta_k = \frac{k - N_p/2}{R}, \qquad k = 0, 1, \ldots, N_p - 1. \tag{3.42}$$

The initial data (3.36) and its discrete Fourier transform are encoded as vectors

$$\boldsymbol{p} := \big(e^{-|p_0|}, \ldots, e^{-|p_{N_p-1}|}\big)^\top, \qquad \boldsymbol{\eta} := \left(\frac{2}{\eta_0^2 + 1}, \ldots, \frac{2}{\eta_{N_p-1}^2 + 1}\right)^\top, \tag{3.43}$$

so that the discretised initial state in the extended space is $v(0) = u(0) \otimes \boldsymbol{p}$ and $\widehat{v}(0) = \mathcal{F}_p[v(0)] = u_0 \otimes \boldsymbol{\eta}$.

Remark: 6. *Since $e^{-|p|}$ belongs to $C^0(\mathbb{R}) \setminus C^1(\mathbb{R})$, the convergence of the discrete Fourier approximation is first-order in Δp. More precisely, the relative discretisation error satisfies*

$$\frac{\|M_{\geq 0} v(T) - u(T) \otimes \boldsymbol{p}_{\geq 0}\|}{\|u(T)\| \, \|\boldsymbol{p}_{\geq 0}\|} = \mathcal{O}\left(\frac{\pi R}{N_p} + e^{-\pi R}\right), \tag{3.44}$$

where $\boldsymbol{p}_{\geq 0} := \sum_{p_k \geq 0} e^{-p_k} |k\rangle$. To achieve a target precision ε one must therefore choose

$$R = \mathcal{O}(\log(1/\varepsilon)), \qquad N_p = \mathcal{O}(R/\varepsilon) = \widetilde{\mathcal{O}}(1/\varepsilon). \tag{3.45}$$

The regularity obstruction in $p = 0$ can be alleviated by replacing $e^{-|p|}$ with a C^k-smooth extension $g \in C^k(\mathbb{R})$ that agrees with e^{-p} on $(0, +\infty)$, yielding the improved bound $N_p = \widetilde{\mathcal{O}}(1/\varepsilon^{1/(k+1)})$ and a corresponding reduction in gate complexity.

3.2.c The Schrödingerisation Pipeline

We now describe the full quantum algorithm in coordinate-free terms. Let $n = n_x + n_p$ be the total qubit count, with the first n_x qubits encoding the spatial degrees of freedom and the last n_p qubits encoding the p-register.

Step 1 **State preparation** Encode $\widehat{v}(0) = u_0 \otimes \boldsymbol{\eta}$ into the quantum register as $|\widehat{v}(0)\rangle = |u_0\rangle \otimes |\boldsymbol{\eta}\rangle$, where $|u_0\rangle = u_0/\|u_0\|$ and $|\boldsymbol{\eta}\rangle = \boldsymbol{\eta}/\|\boldsymbol{\eta}\|$. In practice this is realized applying quantum Fourier transform to the state $|u_0\rangle \otimes |\boldsymbol{p}\rangle$:

$$\mathrm{QFT}_p\big(|u_0\rangle \otimes |\boldsymbol{p}\rangle\big) = |u_0\rangle \otimes |\boldsymbol{\eta}\rangle \, . \tag{3.46}$$

Step 2 **Hamiltonian simulation.** Apply the approximate time-evolution operator $V^r(\tau)$, where $V(\tau)$ is the first-order Lie–Trotter–Suzuki approximation to $e^{iH\tau}$ and $r = T/\tau$ is the number of Trotter steps. Each application of $V(\tau)$ realises one time step of the discretised Schrödinger equation (3.39).

Step 3 **Inverse QFT.** Apply QFT_p^{-1} to the p-register, mapping the Fourier-domain state $|\widehat{v}_D(T)\rangle$ back to the physical-space state $|v_D(T)\rangle$.

Step 4 **Post-selection.** Apply the measurement $M_{\geq 0} = \sum_{p_k \geq 0} I^{\otimes dn_x} \otimes |k\rangle\langle k|$, retaining only the $p_k \geq 0$ component. The post-selected state is proportional to $|u(T)\rangle \, |k\rangle$ for $p_k \approx 0$.

The overall circuit is depicted schematically in Figure 3.2.

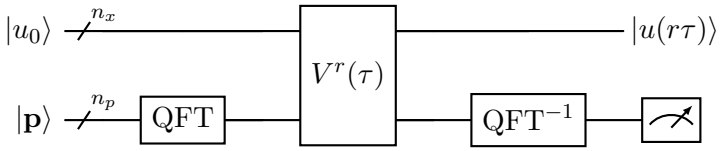


Figure 3.2: Schematic quantum circuit implementing the Schrödingerisation pipeline. The spatial register (top, n_x qubits) and the ancilla p-register (bottom, n_p qubits) interact only through the joint unitary $V^r(\tau)$. The post-selection measurement $M_{\geq 0}$ retains only the $p_k \geq 0$ component of the output.

4 Quantum Circuits for the Heat Equation

We specialize the general Schrödingerisation framework of Chapter 3 to the d-dimensional heat equation. The heat equation is the archetypal second-order parabolic PDE: its generator $a\Delta$ is self-adjoint and non-positive. The Schrödingerisation lift transforms it into a unitary problem in one higher dimension, and the central-difference discretizations of the laplacian admits a strikingly clean decomposition in terms of the shift operators of Section 3.1.

Section 4.1 establishes the continuous and discrete problem. Section 4.2 derives the discretized Hamiltonian H_{heat} and establishes its spectral properties. Section 4.3 constructs, gate by gate, the full circuit hierarchy $W_j \to V_0 \to \tilde{V}_0 \to V_{\text{heat}}$. Section 4.4 contains the rigorous Trotter error analysis that underpins the complexity theorems of Chapter 6.

4.1 Problem Formulation and Discretization

4.1.a The Continuous Problem

Consider the initial-boundary value problem

$$\begin{cases} \partial_t u(t,x) = a\,\Delta u(t,x), & (t,x) \in (0,T] \times \Omega, \\ u(0,x) = f(x), & x \in \Omega, \end{cases} \tag{4.1}$$

where the domain $\Omega \subset \mathbb{R}^d$, the diffusivity constant is greater than zero $a > 0$, and $f \in H_0^1(\Omega)$.

4.1.b Spatial Discretisation

On each spatial coordinate axis, the domain $[0, L]$ is subdivided into $N_x + 1 = 2^{n_x} + 1$ intervals of length $h = L/(N_x + 1)$ where $x_j = jh$, $j = 1, \ldots, N_x$ are the endpoints of each interval. The Dirichlet boundary conditions are imposed restricting to the interior nodes, giving the state vector $u(t) \in \mathbb{R}^{N_x^d}$ with $N_x = 2^{n_x}$. The d-dimensional Laplacian is approximated using the standard second-order central-difference scheme:

$$\frac{\partial^2 u}{\partial x^2}(x_m, t) \approx \frac{u(x_{m+1}, t) - 2u(x_m, t) + u(x_{m-1}, t)}{h^2}. \tag{4.2}$$

In our case the sheme is apllyied via the operator:

$$D_D^{\Delta} = \frac{S^- + S^+ - 2\, I^{\otimes n_x}}{h^2}, \tag{4.3}$$

where S^-, S^+ are the shift operators of Definition 6, so that the semi-discrete system can be written as:

$$\dot{u}(t) = aD_D^{\Delta}\, u(t), \qquad u(0) = u_0. \tag{4.4}$$

Observe that the matrix aD_D^{Δ} is symmetric and negative semi-definite, this implies that the dissipative character of the continuous problem also holds in the discrete system.

4.2 Schrödingerisation and the Discrete Hamiltonian

As mentioned in 5 the decomposition (3.37) gives $H_1 = aD_D^{\Delta}$ and $H_2 = 0$, and the Hamiltonian (3.39) reduces to $H(\eta) = \eta\, aD_D^{\Delta}$. Writing the discretised diagonal operator $D_\eta := \mathrm{diag}(\eta_0, \ldots, \eta_{N_p-1})$, the full $(dn_x + n_p)$-qubit Hamiltonian is

$$H_{\mathrm{heat}} := a \sum_{\alpha=1}^{d} (D_D^{\Delta})_\alpha \otimes D_\eta = \sum_{k=0}^{N_p-1} \left(k - \frac{N_p}{2}\right) \sum_{\alpha=1}^{d} (H_0)_\alpha \otimes |k\rangle\langle k|\,, \tag{4.5}$$

where we have used $\eta_k = (k - N_p/2)/R$ and defined the one-dimensional building-block Hamiltonian

$$H_0 := \gamma_0 \left(\sum_{j=1}^{n_x} (s_j^- + s_j^+) - 2\, I^{\otimes n_x} \right), \qquad \gamma_0 := \frac{a}{h^2 R}. \tag{4.6}$$

The subscript notation $(\cdot)_\alpha := I^{\otimes(d-\alpha)n_x} \otimes \cdot \otimes I^{\otimes(\alpha-1)n_x}$ embeds a single-axis operator into the full d-dimensional register.

Proposition 2. *The operator H_{heat} defined in (4.5) is self-adjoint on $\mathscr{H}_{dn_x+n_p}$.*

Proof. Each summand $(H_0)_\alpha \otimes |k\rangle\langle k|$ is the tensor product of two self-adjoint operators: $H_0 = \gamma_0(D_D^{\Delta} + 2I)$ is self-adjoint as a real symmetric matrix, and $|k\rangle\langle k|$ is an orthogonal projector. A finite sum of self-adjoint operators is self-adjoint. □

A direct consequence of Proposition 2 is that the propagator $e^{iH_{\mathrm{heat}}t}$ is unitary for all t, confirming the applicability of Hamiltonian simulation algorithms.

Block-diagonal structure. The representation (4.5) reveals that H_{heat} is block-diagonal in the p-register basis: the k-th diagonal block is $(k - N_p/2)\sum_\alpha (H_0)_\alpha$. This structure is exploited in the circuit construction of Section 4.3 and yields the identity

$$U_{\mathrm{heat}}(\tau) := e^{iH_{\mathrm{heat}}\tau} = \sum_{k=0}^{N_p-1} \tilde{U}_0^{k-N_p/2}(\tau) \otimes |k\rangle\langle k|\,, \tag{4.7}$$

where $U_0(\tau) := e^{iH_0\tau}$ and $\tilde{U}_0(\tau) := \prod_{\alpha=1}^{d} (U_0(\tau))_\alpha$. In orther to prove the identity (4.7) let first have a look at the two following properties.

Lemma 4. *The following equalities hold:*

$$\begin{aligned} \exp\left(\sum_k A_k \otimes |k\rangle\langle k|\right) &= \sum_k \exp(A_k) \otimes |k\rangle\langle k|, \\ \exp\left(\sum_\alpha (A)_\alpha\right) &= \prod_\alpha (\exp(A))_\alpha, \end{aligned} \tag{4.8}$$

Proof. Recall that $|k\rangle\langle k|$ are orthogonal, so $(|k\rangle\langle k| \cdot |s\rangle\langle s| = \delta_{sk}|k\rangle\langle k|)$.
So if we define X as $X = \sum_k A_k \otimes |k\rangle\langle k|$ we than have that

$$X^n = \sum_k A_k^n \otimes |k\rangle\langle k|$$

and so

$$\begin{aligned} \exp(X) = \sum_n \frac{1}{n!}(X^n) &= \sum_n \sum_k \frac{1}{n!} A_k^n \otimes |k\rangle\langle k| \\ &= \sum_k \left(\sum_n \frac{A_k^n}{n!}\right) \otimes |k\rangle\langle k| \\ &= \sum_k \exp(A_k) \otimes |k\rangle\langle k| \end{aligned}$$

This conclude the first part of the proof.
If, insted, we consider the operator defined as $(A)_\alpha := I^{\otimes d-\alpha} \otimes A \otimes I^{\otimes \alpha-1}$ Recall that given two operators X, Y the identity $e^{X+Y} = e^X e^Y$ holds iff X and$Y$ commute. Due to the definition is trivial that the operators $(A)_\alpha$ commutes. □

Now the identity (4.7) is proven as follow.

$$\begin{aligned} U_{\text{heat}}(\tau) := \exp(i\mathbf{H}_{\text{heat}}\tau) &= \exp\left(i\tau \sum_{k=0}^{N_p-1} \left(k - \frac{N_p}{2}\right) \sum_{\alpha=1}^{d} (\mathbf{H}_0)_\alpha \otimes |k\rangle\langle k|\right) \\ &= \sum_{k=0}^{N_p-1} \exp\left(i\tau \left(k - \frac{N_p}{2}\right) \sum_{\alpha=1}^{d} (\mathbf{H}_0)_\alpha\right) \otimes |k\rangle\langle k| \\ &= \sum_{k=0}^{N_p-1} \left(\exp\left(i\tau \sum_{\alpha=1}^{d} (\mathbf{H}_0)_\alpha\right)\right)^{k-N_p/2} \otimes |k\rangle\langle k| \\ &= \sum_{k=0}^{N_p-1} \left(\prod_{\alpha=1}^{d} \exp\left(i\tau (\mathbf{H}_0)_\alpha\right)\right)^{k-N_p/2} \otimes |k\rangle\langle k| \\ &= \sum_{k=0}^{N_p-1} \left(\prod_{\alpha=1}^{d} (U_\alpha(\tau))_\alpha\right)^{k-N_p/2} \otimes |k\rangle\langle k| \\ &= \sum_{k=0}^{N_p-1} \bar{U}_0^{k-N_p/2}(\tau) \otimes |k\rangle\langle k|. \end{aligned} \tag{4.9}$$

4.3 Circuit Construction

We move forward translating the operator identity (4.7) into an explicit quantum circuit. The construction proceeds through the construction of four successively more complex unitary operators. In order to improve readability, we introduce the following notation.

Notation: 1. *We denote by V any first-order Lie–Trotter–Suzuki approximation of the corresponding exact propagator U, and use, consistently with the notation used above, the tilde to indicate the d-dimensional tensor product: $\tilde{V}_0(\tau) := \prod_{\alpha=1}^{d}(V_0(\tau))_\alpha$.*

4.3.a The Single-Mode Operator $V_0(\tau)$

The building block of the entire construction is the approximate evolution generated by H_0. Applying the first-order LTS approximation

$$\begin{aligned} U_0(\tau) &= \exp\left(i\gamma_0\tau \sum_{j=1}^{n_x}(s_j^- + s_j^+) - 2i\gamma_0\tau I^{\otimes n_x} \right) \\ &\approx \exp(-2i\gamma_0\tau) \prod_{j=1}^{n_x} \exp\left(i\gamma_0\tau(s_j^- + s_j^+) \right) \\ &=: V_0(\tau). \end{aligned}$$

We can rewrite this operator as:

$$V_0(\tau) := e^{-2i\gamma_0\tau} \prod_{j=1}^{n_x} \exp\big(i\gamma_0\tau(s_j^- + s_j^+)\big) = \mathrm{Ph}(-2\gamma_0\tau) \prod_{j=1}^{n_x} I^{\otimes(n_x-j)} \otimes W_j(\gamma_0\tau, 0), \quad (4.10)$$

where $\mathrm{Ph}(\theta) := e^{i\theta} I^{\otimes n_x}$ is the global phase gate and the last equality invokes Lemma 2 with $\lambda = 0$. The operators $\{s_j^- + s_j^+\}_{j=1}^{n_x}$ mutually commute up to terms of operator norm $\mathcal{O}(\gamma_0^2\tau^2)$, which is precisely the Trotter error quantified in Lemma 5.

The circuit realising $V_0(\tau)$ consists of the global phase gate followed by the sequence of $W_j(\gamma_0\tau, 0)$ built as in Lemma 2, each of which acts non-trivially only on qubits $1, \ldots, j$ and is otherwise the identity.

4.3.b The Multi-dimensional Operator $\tilde{V}_0(\tau)$

The d-dimensional analogue is obtained by tensoring:

$$\tilde{V}_0(\tau) := \prod_{\alpha=1}^{d} (V_0(\tau))_\alpha, \quad (4.11)$$

implemented by running d independent copies of the V_0 circuit on each block of n_x spatial qubits. Since the blocks act on disjoint tensor factors they commute exactly, so no further Trotter error is introduced in the transition from V_0 to $\tilde{V}_0$.

4.3.c The Full Time-evolution Operator $V_{\text{heat}}(\tau)$

We now have all the building block necessary in order to construct $V_{\text{heat}}(\tau)$ Substituting the approximation $\tilde{V}_0$ for $\tilde{U}_0$ in the factorisation (4.7) defines

$$V_{\text{heat}}(\tau) := \sum_{k=0}^{N_p-1} \tilde{V}_0^{k-N_p/2}(\tau) \otimes |k\rangle\langle k| = \left(\tilde{V}_0^{-N_p/2}(\tau) \otimes I^{\otimes n_p}\right) \sum_{k=0}^{N_p-1} \tilde{V}_0^{k}(\tau) \otimes |k\rangle\langle k| \,. \quad (4.12)$$

The sum $\sum_k \tilde{V}_0^k(\tau) \otimes |k\rangle\langle k|$ is implemented via Lemma 3: writing k in is binary representation as $\sum_{m=0}^{n_p-1} k_m 2^m$, it decomposes as

$$\sum_{k=0}^{N_p-1} \tilde{V}_0^{k}(\tau) \otimes |k\rangle\langle k| = \prod_{m=0}^{n_p-1}{}' \left(c\text{-}\tilde{V}_0^{2^m}(\tau)\right), \quad (4.13)$$

where $c\text{-}\tilde{V}_0^{2^m}(\tau)$ denotes the controlled-power gate acting on the spatial register, controlled by the m-th qubit of the p-register. This requires $n_p = \log_2 N_p$ controlled-power applications in lieu of N_p direct applications, yielding the quadratic improvement in N_p noted in Remark 4.

The complete Schrödingerisation circuit iterates $V_{\text{heat}}(\tau)$ a total of $r = T/\tau$ times.

4.4 Trotter Error Analysis

The following lemma is the key estimate relating the exact and approximate propagators.

Lemma 5 (Trotter error for V_{heat})**.** *Let H_{heat} and $V_{\text{heat}}(\tau)$ be as in (4.5) and (4.12), and let d denotes the spatial dimension, $N_p = 2^{n_p}$: and $N_x = 2^{n_x}$ represent the number of grid points for the variables p and x, respectively. Then for any $\tau > 0$,*

$$\|U_{\text{heat}}(\tau) - V_{\text{heat}}(\tau)\| \le \frac{d\, N_p\, \gamma_0^2\, \tau^2\, (n_x - 1)}{4}, \quad (4.14)$$

and for r Trotter steps of size $r = T/\tau$,

$$\|U_{\text{heat}}(T) - V_{\text{heat}}^r(\tau)\| \le \frac{d\, N_p\, \gamma_0^2\, T^2\, (n_x - 1)}{4r}. \quad (4.15)$$

Proof. We proceed in three steps.

Step 1: Error for V_0. The operators $s_j^- + s_j^+$ for $j = 1, \ldots, n_x$ satisfy $\left\|s_j^{\pm}\right\| = 1$ (cf. Lemma 1). The standard commutator bound for the first-order LTS formula gives

$$\|U_0(\tau) - V_0(\tau)\| \le \frac{\gamma_0^2 \tau^2}{2} \sum_{j=1}^{n_x} \sum_{j'>j} \left\| \left[s_j^- + s_j^+, s_{j'}^- + s_{j'}^+ \right] \right\|. \quad (4.16)$$

The commutator calculations of ([14], Appendix B.1) establish that:

$$\|U_0(\tau) - V_0(\tau)\| \le \frac{\gamma_0^2 \tau^2}{2} \sum_{j=1}^{n_x} \sum_{j'>j} \left\| \left[s_j^- + s_j^+, s_{j'}^- + s_{j'}^+ \right] \right\| \le \frac{\gamma_0^2 \tau^2 (n_x - 1)}{2}. \quad (4.17)$$

Step 2: Propagation to V_{heat}. Since $\tilde{V}_0(\tau)$ and $\tilde{U}_0(\tau)$ are unitary, and the block-diagonal structure of (4.7) allows one to bound the deviation block by block obtaning:

$$\begin{aligned}
&\|U_{\text{heat}}(\tau) - V_{\text{heat}}(\tau)\| \\
&= \left\| \sum_{k=0}^{N_p-1} \left[\left(\prod_{\alpha=1}^{d} (U_0(\tau))_\alpha \right)^{k-N_p/2} - \left(\prod_{\alpha=1}^{d} (V_0(\tau))_\alpha \right)^{k-N_p/2} \right] \otimes |k\rangle\langle k| \right\| \\
&= \max_{0\le k\le N_p-1} \left\| \left(\prod_{\alpha=1}^{d} (U_0(\tau))_\alpha \right)^{k-N_p/2} - \left(\prod_{\alpha=1}^{d} (V_0(\tau))_\alpha \right)^{k-N_p/2} \right\| \\
&\le \max_{0\le k\le N_p-1} \left| k - \frac{N_p}{2} \right| \left\| \prod_{\alpha=1}^{d} (U_0(\tau))_\alpha - \prod_{\alpha=1}^{d} (V_0(\tau))_\alpha \right\| \\
&\le \frac{dN_p}{2} \|U_0(\tau) - V_0(\tau)\| \\
&\le \frac{dN_p \gamma_0^2 \tau^2 (n_x - 1)}{4}.
\end{aligned} \tag{4.18}$$

Step 3: Accumulation over r steps. Having proved 4.14 4.15 simplu follows as:

$$\begin{aligned}
&\|U_{\text{heat}}(T) - V_{\text{heat}}^r(\tau)\| \\
\le& r \|U_{\text{heat}}(\tau) - V_{\text{heat}}(\tau)\| \\
\le& \frac{dN_p \gamma_0^2 T^2 (n_x - 1)}{4r}.
\end{aligned} \tag{4.19}$$

□

Remark: 7 (Required number of Trotter steps). *To achieve $\|U_{\text{heat}}(T) - V_{\text{heat}}^r(\tau)\| \le \varepsilon'$, Lemma 5 requires*

$$r \ge \frac{dN_p \gamma_0^2 T^2 (n_x - 1)}{4\varepsilon'}. \tag{4.20}$$

The dependence on n_x is linear (not quadratic), reflecting the locality of the commutators $\left[s_j^{\pm}, s_{j'}^{\pm} \right]$.

5 Quantum Circuits for the Advection Equation

The advection equation presents a distinct challenge: as mentioned in 5, unlike the heat operator, the upwind-discretised advection operator A is neither symmetric nor skew-symmetric, so in the decomposition both the Hermitian part H_1 and the skew-Hermitian part H_2 are non-trivial. This requires a more complex circuit structure in which the two contributions are handled by two separate unitary building blocks V_1 and V_2, whose tensor-product structure over spatial dimensions and whose controlled-power combination over the p-register must be carefully arranged.

A further motivation for the treatment developed here is the inadequacy of the central-difference approach. Sato et al. [21] employ central differences to preserve the Schrödinger's structure of the advection operator; however, the central scheme introduces dispersive errors that generate numerical oscillations near discontinuities of the solution, as illustrated in their own numerical results. The upwind scheme is dissipative rather than dispersive, suppressing these oscillations at the cost of first-order accuracy in h, and it is precisely this dissipative character that necessitates Schrödingerisation.

5.1 Problem Formulation and Upwind Discretisation

5.1.a The Continuous Problem

Let $\mathbf{a} = (a_1, \dots, a_d)^\top \in \mathbb{R}^d$ be the constant advection velocity. We consider the initial-value problem

$$\begin{cases} \partial_t u(t,x) = \mathbf{a} \cdot \nabla u(t,x) = \sum_{\alpha=1}^{d} a_\alpha \partial_{x_\alpha} u(t,x), \quad (t,x) \in (0,T] \times \Omega^d, \\ u(0,x) = f(x), \end{cases} \tag{5.1}$$

on the d-dimensional domain $\Omega^d = (0,L)^d$ with periodic boundary conditions. As known the exact solution is the pure transport $u(t,x) = f(x + \mathbf{a}t)$, which preserves the L^2-norm.

5.1.b Upwind Discretization

In order to discretize the problem consider the uniform grid of $N_x = 2^{n_x}$ points per dimension with mesh size $h = L/N_x$. On such grid the upwind scheme approximates

$a_\alpha \partial_{x_\alpha} u$ via:

$$a_\alpha \partial_{x_\alpha} u \approx \begin{cases} a_\alpha (D_P^+)_\alpha \, u, & a_\alpha > 0, \\ a_\alpha (D_P^-)_\alpha \, u, & a_\alpha < 0, \end{cases} \tag{5.2}$$

where D_P^+ and D_P^- are the periodic forward and backward difference operators of Subsection 3.1.a. Without loss of generality, we can rearrange the coordinates in such a way that $a_\alpha > 0$ for $\alpha = 1, \ldots, d_0$ and $a_\alpha < 0$ for $\alpha = d_0 + 1, \ldots, d$. The semi-discrete system will then be:

$$\dot{u}(t) = A\, u(t), \qquad A := \sum_{\alpha=1}^{d_0} a_\alpha (D_P^+)_\alpha + \sum_{\alpha=d_0+1}^{d} a_\alpha (D_P^-)_\alpha. \tag{5.3}$$

Observe that the matrix A is real but non-symmetric: its symmetric component $H_1 = \dfrac{(A + A^\dagger)}{2}$ depicts the diffusive character of the upwind scheme, while its skew-symmetric component $iH_2 = \dfrac{(A - A^\dagger)}{2}$ encodes the conservative transport.

Direct computation using the identities $D_P^+ = \dfrac{(D_P^\pm) + (D_P^\Delta)}{2}$ and $D_P^- = \dfrac{(D_P^\pm) - (D_P^\Delta)}{2}$ yields:

$$H_1 = \sum_{\alpha=1}^{d} \frac{|a_\alpha|}{2} h (D_P^\Delta)_\alpha, \qquad H_2 = \sum_{\alpha=1}^{d} \frac{a_\alpha}{2h} \cdot \frac{1}{i} (S_\alpha^- - S_\alpha^+ - \sigma_{01,\alpha}^{\otimes n_x} + \sigma_{10,\alpha}^{\otimes n_x}), \tag{5.4}$$

showing that H_1 is a positive diffusion operator and H_2 is a central-difference-like transport operator.

5.2 Schrödingerization and the Discrete Hamiltonian

Applying the warped phase transformation of Section 3.2 to our new system $\dot{u} = Au$ with $A = H_1 + iH_2$ and relying on the Fourier space in p we obtain the Schrödinger equation:

$$\partial_t \widehat{v}(t, \eta) = i\, H_{\text{adv}}(\eta)\, \widehat{v}(t, \eta), \qquad H_{\text{adv}}(\eta) := \eta\, H_1 + H_2, \tag{5.5}$$

which is self-adjoint for every $\eta \in \mathbb{R}$ since $H_1 = H_1^\dagger$ and $H_2 = H_2^\dagger$. Replacing η by the discrete grid (3.42), the full discretized Hamiltonian is

$$\begin{aligned} H_{\text{adv}} :=& H_1 \otimes D_\eta + H_2 \otimes I^{\otimes n_p} \\ =& \sum_{k=0}^{N_p - 1} \left(k - \frac{N_p}{2} \right) \sum_{\alpha=1}^{d} |a_\alpha| (H_1)_\alpha \otimes |k\rangle\langle k| + \sum_{\alpha=1}^{d} a_\alpha (H_2)_\alpha \otimes I^{\otimes n_p}, \end{aligned} \tag{5.6}$$

where we define, with $\gamma_1 := 1/(2hR)$ and $\gamma_2 := 1/(2h)$:

$$
\begin{aligned}
H_1 &:= H_1^{(1)} + H_1^{(2)}, \\
H_1^{(1)} := \gamma_1\big(\sigma_{01}^{\otimes n_x} + \sigma_{10}^{\otimes n_x}\big), \quad H_1^{(2)} &:= \gamma_1 \sum_{j=1}^{n_x}(s_j^- + s_j^+) - 2\gamma_1\, I^{\otimes n_x},
\end{aligned} \tag{5.7}
$$

$$
\begin{aligned}
H_2 &:= H_2^{(1)} + H_2^{(2)}, \\
H_2^{(1)} := -i\gamma_2\big(-\sigma_{01}^{\otimes n_x} + \sigma_{10}^{\otimes n_x}\big), \quad H_2^{(2)} &:= -i\gamma_2 \sum_{j=1}^{n_x}(s_j^- - s_j^+).
\end{aligned} \tag{5.8}
$$

The superscripts (1) and (2) distinguish the *boundary terms* (arising from the periodic wrap-around $\sigma_{01}^{\otimes n_x}$, $\sigma_{10}^{\otimes n_x}$) from the *interior terms* (involving the local shift operators $s_j^{\pm}$). Such distinction between $H_1 = H_1^{(1)} + H_1^{(2)}$ and $H_2 = H_2^{(1)} + H_2^{(2)}$ is not merely for notational convenience, the two cases require distinct circuit implementations. The interior terms $H_1^{(2)}, H_2^{(2)}$ are implemented via a W_j-type circuits, as in the heat equation, whereas the boundary terms $H_1^{(1)}, H_2^{(1)}$ are handled by different circuits described in Subection 5.3.b.

5.3 Circuit Construction

5.3.a Factorisation of the Propagator

Cloning the construction we made for Section 4.3 we introduce:

$$
\begin{aligned}
U_1(\tau) &:= e^{iH_1\tau}, & U_2(\tau) &:= e^{iH_2\tau}, \\
\tilde{U}_1(\tau) &:= \prod_{\alpha=1}^{d}(U_1(|a_\alpha|\tau))_\alpha, & \tilde{U}_2(\tau) &:= \prod_{\alpha=1}^{d}(U_2(a_\alpha\tau))_\alpha.
\end{aligned} \tag{5.9}
$$

The first-order LTS decomposition allows us to approximate the time evolution operator $U_{\text{adv}}(\tau)$ as follows:

$$
\begin{aligned}
U_{\text{adv}}(\tau) &:= \exp(i\mathbf{H}_{\text{adv}}\tau) \\
&= \exp\left(i\tau\left(\sum_{k=0}^{N_p-1}\left(k-\frac{N_p}{2}\right)\sum_{\alpha=1}^{d}|a_\alpha|\,(\mathbf{H}_1)_\alpha \otimes |k\rangle\langle k| + \sum_{\alpha=1}^{d} a_\alpha\,(\mathbf{H}_2)_\alpha \otimes I^{\otimes n_p}\right)\right) \\
&\approx \exp\left(i\tau\sum_{\alpha=1}^{d} a_\alpha\,(\mathbf{H}_2)_\alpha \otimes I^{\otimes n_p}\right)\exp\left(i\tau\sum_{k=0}^{N_p-1}\left(k-\frac{N_p}{2}\right)\sum_{\alpha=1}^{d}|a_\alpha|\,(\mathbf{H}_1)_\alpha \otimes |k\rangle\langle k|\right) \\
&= \left(\prod_{\alpha=1}^{d}\exp\left(ia_\alpha\tau\,(\mathbf{H}_2)_\alpha\right)\otimes I^{\otimes n_p}\right)\sum_{k=0}^{N_p-1}\left(\prod_{\alpha=1}^{d}\exp\left(i|a_\alpha|\tau\,(\mathbf{H}_1)_\alpha\right)\right)^{k-N_p/2} \otimes |k\rangle\langle k| \\
&= \left(\prod_{\alpha=1}^{d}(U_2(a_\alpha\tau))_\alpha \otimes I^{\otimes n_p}\right)\sum_{k=0}^{N_p-1}\left(\prod_{\alpha=1}^{d}(U_1(|a_\alpha|\tau))_\alpha\right)^{k-N_p/2} \otimes |k\rangle\langle k| \\
&= \left(\tilde{U}_2(\tau)\otimes I^{\otimes n_p}\right)\sum_{k=0}^{N_p-1}\tilde{U}_1^{k-N_p/2}(\tau)\otimes|k\rangle\langle k|,
\end{aligned} \tag{5.10}
$$

where the first equality relies on 5.6, third equality holds due to the results

of the Lemma 4, meanwhile the third and the fourth are given directly using the definitions 5.9. We will show in Lemma 6 how the approximation error is controlled by the commutator $\|[H_1 \otimes D_\eta, H_2 \otimes I^{\otimes n_p}]\|$.
Replacing U_1, U_2 by their LTS approximations V_1, V_2 (constructed in Section 5.3.d) then define:

$$V_{\text{adv}}(\tau) := \left(\tilde{V}_2(\tau) \otimes I^{\otimes n_p}\right) \sum_{k=0}^{N_p-1} \tilde{V}_1^{k-N_p/2}(\tau) \otimes |k\rangle\langle k| \,, \tag{5.11}$$

that can be implemented via the controlled-power technique of Lemma 3.

5.3.b Circuits for the Boundary Terms

The operators $H_1^{(1)}$ and $H_2^{(1)}$ satisfy the hypothesis of Lemma 1, where $|a\rangle = |0\rangle^{\otimes n_x}$ and $|b\rangle = |1\rangle^{\otimes n_x}$. Therefore their time-evolution operators are:

$$\begin{aligned} U_1^{(1)}(\tau) &= \exp\left(i\gamma_1\tau(\sigma_{01}^{\otimes n_x} + \sigma_{10}^{\otimes n_x})\right) = B^{(1)} \cdot \mathrm{CRZ}^{n_x}_{1,\dots,n_x-1}(-2\gamma_1\tau) \cdot \left(B^{(1)}\right)^\dagger, \\ U_2^{(1)}(\tau) &= \exp\left(\gamma_2\tau(-\sigma_{01}^{\otimes n_x} + \sigma_{10}^{\otimes n_x})\right) = B^{(2)} \cdot \mathrm{CRZ}^{n_x}_{1,\dots,n_x-1}(-2\gamma_2\tau) \cdot \left(B^{(2)}\right)^\dagger, \end{aligned} \tag{5.12}$$

where the basis-change unitaries are given by

$$\begin{aligned} B^{(1)} &:= \prod_{m=1}^{n_x-1} \mathrm{CNOT}^{n_x}_m H_{n_x} \prod_{m=1}^{n_x-1} X_m = B_{n_x}(0) \prod_{m=1}^{n_x-1} X_m, \\ B^{(2)} &:= \prod_{m=1}^{n_x-1} \mathrm{CNOT}^{n_x}_m P_{n_x}\left(-\frac{\pi}{2}\right) H_{n_x} \prod_{m=1}^{n_x-1} X_m = B_{n_x}\left(\frac{\pi}{2}\right) \prod_{m=1}^{n_x-1} X_m. \end{aligned} \tag{5.13}$$

Here $B_{n_x}(\lambda)$ is the Bell-basis construction of Lemma 2, and the X-gate chain maps $|0\rangle^{\otimes(n_x-1)} \to |1\rangle^{\otimes(n_x-1)}$, aligning the all-zeros and all-ones states with the standard Bell-basis conventions.

5.3.c Circuits for the Interior Terms

The interior Hamiltonians $H_1^{(2)}$ and $H_2^{(2)}$ have the same structure as H_0 in the heat equation, but with different phase parameters. Lemma 2 with $\lambda = 0$ and $\lambda = -\pi/2$ respectively gives

$$U_1^{(2)}(\tau) \approx V_1^{(2)}(\tau) := \mathrm{Ph}(-2\gamma_1\tau) \prod_{j=1}^{n_x} I^{\otimes(n_x-j)} \otimes W_j(\gamma_1\tau, 0), \tag{5.14}$$

$$U_2^{(2)}(\tau) \approx V_2^{(2)}(\tau) := \prod_{j=1}^{n_x} I^{\otimes(n_x-j)} \otimes W_j(\gamma_2\tau, -\pi/2). \tag{5.15}$$

Notice that the phase shift $\lambda = -\pi/2$ in (5.15) accounts for the factor of $-i$ in the definition of $H_2^{(2)}$, and it is realized by the Phase gate $P(-\pi/2) = P_j(-\pi/2)$ within each W_j gadget.

5.3.d Assembling $V_1(\tau)$ and $V_2(\tau)$

Combining boundary and interior circuits via a further LTS step:

$$V_1(\tau) := U_1^{(1)}(\tau)\, V_1^{(2)}(\tau), \tag{5.16}$$

$$V_2(\tau) := U_2^{(1)}(\tau)\, V_2^{(2)}(\tau). \tag{5.17}$$

The LTS error incurred in replacing $U_1(\tau)$ by $V_1(\tau)$ involves the commutator $\left\| \left[H_1^{(1)}, H_1^{(2)} \right] \right\|$, which is bounded. Analogously for V_2.

Tensoring over the spatial dimensions we obtain

$$\tilde{V}_1(\tau) := \prod_{\alpha=1}^{d} \big(V_1(|a_\alpha|\tau)\big)_\alpha, \qquad \tilde{V}_2(\tau) := \prod_{\alpha=1}^{d} \big(V_2(a_\alpha \tau)\big)_\alpha, \tag{5.18}$$

each implemented by running parallel clones of V_1 (resp. V_2) on the d spatial qubit blocks, with axis-dependent time parameters $|a_\alpha|\tau$ (resp. $a_\alpha\tau$). Since the blocks act on disjoint tensor factors, no additional Trotter error arises from the tensor-product assembly.

The full circuit for $V_{\text{adv}}(\tau)$ than combines the $\tilde{V}_2$ block with the controlled-power $\sum_k \tilde{V}_1^{k-N_p/2}(\tau) \otimes |k\rangle\langle k|$, implemented exactly as in Subsection 4.3.c via Lemma 3.

5.4 Trotter Error Analysis

Lemma 6 (Trotter error for V_{adv})**.** *Let H_{adv} and $V_{\text{adv}}(\tau)$ be as defined in (5.6) and (5.11). Set $C_{\mathbf{a}} := \sum_{\alpha=1}^{d} a_\alpha^2$. Then for any $\tau > 0$,*

$$\|U_{\text{adv}}(\tau) - V_{\text{adv}}(\tau)\| \le \frac{\tau^2\, n_x\, C_{\mathbf{a}}}{4} \big(N_p\gamma_2^2 + 2N_p\gamma_1\gamma_2 + 2\gamma_1^2\big), \tag{5.19}$$

and for r Trotter steps of size $\tau = T/r$,

$$\|U_{\text{adv}}(T) - V_{\text{adv}}^r(\tau)\| \le \frac{T^2\, n_x\, C_{\mathbf{a}}}{4r} \big(N_p\gamma_2^2 + 2N_p\gamma_1\gamma_2 + 2\gamma_1^2\big). \tag{5.20}$$

Proof. Let's divide the proof in steps.

Step 1:

$$\begin{aligned}
\|U_1(\tau) - V_1(\tau)\| &\le \left\|U_1(\tau) - U_1^{(1)}(\tau)U_1^{(2)}(\tau)\right\| + \left\|U_1^{(1)}(\tau)U_1^{(2)}(\tau) - V_1(\tau)\right\| \\
&\le \frac{\gamma_1^2\tau^2}{2} \left\| \left[\sum_{j=1}^{n_x} (s_j^- + s_j^+), \sigma_{01}^{\otimes n_x} + \sigma_{10}^{\otimes n_x} \right] \right\| + \left\|U_1^{(2)}(\tau) - V_1^{(2)}(\tau)\right\| \\
&\le \frac{\gamma_1^2\tau^2}{2} + \frac{\gamma_1^2\tau^2(n_x-1)}{2} = \frac{\gamma_1^2\tau^2 n_x}{2},
\end{aligned}$$

Step 2:

$$\begin{aligned}\|U_2(\tau) - V_2(\tau)\| &\leq \left\|U_2(\tau) - U_2^{(1)}(\tau)U_2^{(2)}(\tau)\right\| + \left\|U_2^{(1)}(\tau)U_2^{(2)}(\tau) - V_2(\tau)\right\| \\ &\leq \frac{\gamma_2^2\tau^2}{2}\left\|\left[\sum_{j=1}^{n_x}(s_j^- - s_j^+), \sigma_{01}^{\otimes n_x} - \sigma_{10}^{\otimes n_x}\right]\right\| + \left\|U_2^{(2)}(\tau) - V_2^{(2)}(\tau)\right\| \\ &\leq \frac{\gamma_2^2\tau^2}{2} + \frac{\gamma_2^2\tau^2(n_x-1)}{2} = \frac{\gamma_2^2\tau^2 n_x}{2},\end{aligned}$$

Step 3:

$$\begin{aligned}\left\|\tilde{U}_1(\tau) - \tilde{V}_1(\tau)\right\| &\leq \frac{\gamma_1^2\tau^2 n_x}{2}\sum_{\alpha=1}^{d} a_\alpha^2, \\ \left\|\tilde{U}_2(\tau) - \tilde{V}_2(\tau)\right\| &\leq \frac{\gamma_2^2\tau^2 n_x}{2}\sum_{\alpha=1}^{d} a_\alpha^2.\end{aligned}$$

Step 4:

$$\begin{aligned}\left\|\left[\mathbf{A}_1 \otimes D_\eta, \mathbf{A}_2 \otimes I^{\otimes n_p}\right]\right\| &= \left\|\left[\sum_{k=0}^{N_p-1}\left(k - \frac{N_p}{2}\right)\sum_{\alpha=1}^{d}|a_\alpha|\left(\mathbf{H}_1\right)_\alpha \otimes |k\rangle\langle k|, \sum_{\alpha=1}^{d} a_\alpha\left(\mathbf{H}_2\right)_\alpha \otimes I^{\otimes n_p}\right]\right\| \\ &= \left\|\sum_{k=0}^{N_p-1}\left(k - \frac{N_p}{2}\right)\left[\sum_{\alpha=1}^{d}|a_\alpha|\left(\mathbf{H}_1\right)_\alpha, \sum_{\alpha=1}^{d} a_\alpha\left(\mathbf{H}_2\right)_\alpha\right] \otimes |k\rangle\langle k|\right\| \\ &= \left\|\sum_{k=0}^{N_p-1}\left(k - \frac{N_p}{2}\right)\sum_{\alpha=1}^{d}|a_\alpha|a_\alpha\left[\left(\mathbf{H}_1\right)_\alpha, \left(\mathbf{H}_2\right)_\alpha\right] \otimes |k\rangle\langle k|\right\| \\ &\leq \frac{N_p\sum_{\alpha=1}^{d}a_\alpha^2}{2}\|[\mathbf{H}_1, \mathbf{H}_2]\| \\ &\leq N_p\gamma_1\gamma_2 n_x\sum_{\alpha=1}^{d}a_\alpha^2.\end{aligned}$$

Step 5:

$$\begin{aligned}\|U_{\text{adv}}(\tau) - V_{\text{adv}}(\tau)\| &\leq \|U_{\text{adv}}(\tau) - U_*(\tau)\| + \|U_*(\tau) - V_{\text{adv}}(\tau)\| \\ &\leq \frac{\tau^2}{2}\left\|\left[\mathbf{A}_1 \otimes D_\eta, \mathbf{A}_2 \otimes I^{\otimes n_p}\right]\right\| \\ &\quad + \frac{N_p}{2}\left\|\tilde{U}_1(\tau) - \tilde{V}_1(\tau)\right\| + \left\|\tilde{U}_2(\tau) - \tilde{V}_2(\tau)\right\| \\ &\leq \frac{\tau^2 N_p\gamma_1\gamma_2 n_x}{2}\sum_{\alpha=1}^{d}a_\alpha^2 \\ &\quad + \frac{N_p\gamma_1^2\tau^2 n_x}{4}\sum_{\alpha=1}^{d}a_\alpha^2 + \frac{\gamma_1^2\tau^2 n_x}{2}\sum_{\alpha=1}^{d}a_\alpha^2 \\ &\leq \frac{\tau^2 n_x(N_p\gamma_1^2 + 2N_p\gamma_1\gamma_2 + 2\gamma_2^2)}{4}\sum_{\alpha=1}^{d}a_\alpha^2.\end{aligned}$$

This completes the proof for the single step case. As done in Lemma 5 over r steps we obtain:

$$\begin{aligned}\|U_{\text{adv}}(T) - V_{\text{adv}}^r(\tau)\| &\leq r\,\|U_{\text{adv}}(\tau) - V_{\text{adv}}(\tau)\| \\ &\leq \frac{T^2 n_x(N_p\gamma_1^2 + 2N_p\gamma_1\gamma_2 + 2\gamma_2^2)}{4r}\sum_{\alpha=1}^{d}a_\alpha^2.\end{aligned} \tag{5.21}$$

□

Remark: 8 (Comparison with the central-difference approach)**.** *In Sato et al. [21], the advection equation is handled via central differences, for which the semi-discrete operator is purely imaginary (i.e. $H_1 = 0$). The Schrödinger's structure is then preserved exactly at the cost of $\mathcal{O}(h^2)$ dispersive errors. In our upwind formulation, $H_1 \neq 0$ introduces the γ_1-dependent terms in the error bound (5.19); however, the solution is free of spurious oscillations near discontinuities. Moreover, our approach handles general non-Hermitian generators, including operators for which no purely imaginary discretization is available. The factor N_p^2 appearing in the gate complexity of [21] versus the N_p factor in ours (see Remark 9 below) reflects this structural difference.*

Remark: 9 (Required number of Trotter steps for the advection equation)**.** *The bound (5.20) requires, for precision ε',*

$$r \geq \frac{T^2 n_x C_{\mathbf{a}}}{4\varepsilon'} \left(N_p \gamma_2^2 + 2 N_p \gamma_1 \gamma_2 + 2\gamma_1^2 \right). \tag{5.22}$$

Substituting $\gamma_1 = 1/(2hR)$ and $\gamma_2 = 1/(2h)$ and using $N_p = \widetilde{\mathcal{O}}(1/\varepsilon)$, one finds that the dominant term at small h is $\gamma_2^2 N_p = \mathcal{O}(N_p/h^2)$, leading to the gate complexity of the next Chapter.

6 Complexity Analysis and Quantum Advantages

Having established in the previous chapters the theoretical foundations of the Schrödingerisation method and the explicit construction of quantum circuits for both the heat and advection equations, we now turn to a rigorous analysis of the computational resources required by these algorithms. The present chapter serves a double purpose: first, to quantify the gate complexity of the circuits developed in Chapters 4 and 5 in terms of fundamental quantum operations; and second, to compare these quantum complexities against the computational costs of classical algorithms solving the same problems, allowing us to identify the cases in which the resulting quantum algorithms outperform the best known classical methods.

The analysis is organized as follows: Section 6.1 derives the CNOT counts for each building block of the circuit hierarchy. Sections 6.2 and 6.3 assemble these into the total gate complexity for the heat and advection equations respectively, culminating in Theorems 1 and 2. Section 6.4 carries out the classical-versus-quantum comparison and identifies the dimensional thresholds beyond which quantum advantage is rigorously established. Section 6.5 situates the results in the broader literature, with emphasis on the oracle-free character of the present constructions.

The complexity analysis presented here is grounded in the explicit circuit decompositions established in Lemmas 2, and in the sections 4.3 5.3, and follows the methodology introduced by the authors of the original article [14]. Unlike many quantum algorithms for partial differential equations that rely on black-box oracles for matrix operations (whose preparation costs are often omitted from complexity estimates) the algorithms analyzed in this thesis are entirely oracle-free. All operations are decomposed into elementary single-qubit gates and CNOT gates, ensuring that the complexity bounds derived here reflect the true resource requirements of implementation on near-term and future quantum hardware.

Throughout this chapter, we employ the standard asymptotic notation: $\mathcal{O}(\cdot)$ denotes an upper bound up to constant factors, while $\widetilde{\mathcal{O}}(\cdot)$ suppresses logarithmic factors. The parameters n_x and n_p denote, respectively, the number of qubits allocated to the spatial and momentum registers, with corresponding grid sizes $N_x = 2^{n_x}$ and $N_p = 2^{n_p}$. The spatial dimension is denoted by d, and T represents the final simulation time.

6.1 CNOT Counts for the Circuit Building Blocks

We begin our analysis with the heat equation, whose quantum circuit was developed in Section 4.3. Recall that the approximate time-evolution operator $V_{\text{heat}}(\tau)$ is constructed through a first-order Lie-Trotter-Suzuki decomposition of the exact operator $U_{\text{heat}}(\tau) = \exp(iH_{\text{heat}}\tau)$, where the Hamiltonian H_{heat} is given by Equation (4.5). The operator $V_{\text{heat}}(\tau)$ comprises three nested levels of structure: the building block W_j, the single-dimension operator V_0, and the multi-dimensional operator $\widetilde{V}_0$, which allows us to obtain full momentum-dependent operator V_{heat} (4.12) via the controlled-power construction of Lemma 3.

We will apply a bottom-up approach beginning with the fundamental building block and propagating upward to the complete circuit. This approach ensures that every gate in the final circuit is accounted for. The analysis proceeds in four stages: first, we establish the gate count for the fundamental building block W_j; second, we aggregate these counts to obtain the complexity of the single-dimension operator V_0 and its controlled version; third, we propagate these bounds through the multi-dimensional and momentum-dependent structures; and finally, we combine the per-step gate count with the Trotter error analysis to obtain the total complexity for the full simulation.

We systematically derive the CNOT counts for the gate hierarchy $W_j \to V_0 \to V_{\text{heat}}$ and its advection-equation counterpart $W_j \to V_1, V_2 \to V_{\text{adv}}$.

6.1.a Gate Count for the Building Block W_j

Recall from Lemma 2 that

$$W_j(\gamma\tau, \lambda) = B_j(\lambda)\, \mathrm{CRZ}^j_{1,\dots,j-1}(-2\gamma\tau)\, B_j(\lambda)^\dagger,$$

where $B_j(\lambda) = \big(\prod_{m=1}^{j-1} \mathrm{CNOT}^j_m\big) P_j(-\lambda) H_j$. Written in this explicit way it is easy to observe that the construction of the pair $B_j(\lambda)$ and $B_j(\lambda)^\dagger$ requires exactly 4 single-qubit gates (two Hadamard gates and two phase gates) and $2(j-1)$ CNOT gates.
Relying on the results obtained by [22], we can establish the total amount of gates needed in the construction of the whole W_j.

Proposition 3. *For $n_x \geq 3$ and $j \in \{1, \dots, n_x\}$, the gadget $W_j(\gamma\tau, \lambda)$ is implemented with*

$$Q_{W_j} \leq \begin{cases} 0 & j = 1, \\ 2(j-1) + 2 = 4 & j = 2, \\ (16j - 40) + 2(j-1) = 18j - 42 & j \geq 3. \end{cases} \tag{6.1}$$

CNOT gates, together with $\mathcal{O}(j)$ single-qubit gates.

Proof. We have seen how the unitary $B_j(\lambda)$ contributes. The central component of W_j is the multi-controlled rotation $\mathrm{CRZ}_{1,\dots,j-1}(-2\gamma\tau)$, which is a $(j-1)$-controlled RZ gate acting on the j-th qubit. This gate applies a rotation by angle $-2\gamma\tau$ about the Z-axis of the

Bloch sphere, conditioned on all of the first $j-1$ qubits being in the state $|1\rangle$. The multi-controlled R_Z gate with $j-1$ control qubits decomposes into at most $\max(1, 16j-40)$ CNOT gates and $\mathcal{O}(j)$ single-qubit gates for $j \geq 3$ by [22]. For the two special cases that arise we have that: if $j=1$ the gate is a plain R_Z, requiring no CNOT; if $j=2$ the operator W_2 requires a single controlled R_Z, which decomposes into two CNOT gates and three single-qubit rotations (two R_Z gates and one R_X or R_Y gate for the Euler decomposition). Summing the three contributions establishes (6.1). □

Remark: 10. *Despite what we have just seen the authors adopt a slightly coarser but common convention: in relation to the single-qubit gates count they consider only the gates acting as proper single-qubit rotations on the target qubit, considering only two single-qubit gates per W_j (the two Hadamards gates) treating the phase gates and R_Z as part of the controlled structure.*

These bounds, while expressed in terms of the maximum possible gate counts, are tight in the sense that they reflect the actual circuit constructions used in the implementation.

6.1.b Gate Count for the Controlled Operator V_0

We first recall how V_0 is built in (4.10), i.e.

$$V_0(\tau) = \mathrm{Ph}(-2\gamma_0\tau) \prod_{j=1}^{n_x} I^{\otimes(n_x-j)} \otimes W_j(\gamma_0\tau, 0).$$

Notice how the blocks W_j are applied in order of increasing j, with each W_j acting on the first j qubits of the spatial register. Since the supports of consecutive W_j operators overlap (each W_j affects qubit j and all preceding qubits), they must be applied sequentially rather than in parallel. Meanwhile, the global phase gate commutes with all other operations and can be applied at any point in the circuit. It contributes as one single-qubit gate.

Lemma 7 (Gate count for V_0)**.** *The operator $V_0(\tau)$ can be implemented using $2n_x+1$ single-qubit gates and at most $9n_x^2 - 33n_x + 36$ CNOT gates for $n_x \geq 3$.*

Proof. The single-qubit gates count comes from the one phase gate plus the two Hadamard gates for each of the W_j across $j = 1, \ldots, n_x$, obtaining $2n_x + 1$.
For the CNOT gate count, we sum the exact contributions of each W_j. The cases $j=1$ and $j=2$ contribute 0 and 4 CNOTs, respectively. For $j \geq 3$, each W_j contributes at most $18j - 42$ CNOTs. We therefore compute:

$$\begin{aligned} Q_{V_0}^{(\mathrm{CNOT})} &= \sum_{j=3}^{n_x} (18j - 42) + 4 \\ &= 18 \sum_{j=3}^{n_x} j - 42(n_x - 2) + 4 \\ &= 18 \left(\frac{n_x(n_x+1)}{2} - 3 \right) - 42n_x + 84 + 4 \\ &= 9n_x^2 + 9n_x - 54 - 42n_x + 88 \\ &= 9n_x^2 - 33n_x + 36. \end{aligned} \tag{6.2}$$

This completes the proof. □

6.1.c Gate Count for the Controlled Operator c-V_0

The controlled version c-V_0, required for the momentum-dependent operator V_{heat}, is obtained by replacing each gate in V_0 with its controlled variant. In general a controlled single-qubit gate decomposes into at most two CNOTs and single-qubit rotations via standard circuit identities, while a controlled CNOT becomes a Toffoli gate, which requires six CNOTs in standard decompositions using one ancilla qubit. However, we can apply a more efficient direct analysis, which carefully tracks the structure of the controlled-W_j operators obtaining a tighter bound.

Lemma 8 (Gate count for c-V_0)**.** *The controlled operator c-$V_0(\tau)$ can be implemented using most $16n_x^2 - 22n_x + 10$ CNOT gates for $n_x \geq 3$.*

Proof. The controlled-V_0 consists of a controlled phase gate and n_x controlled-W_j operators. Each controlled-W_j contains: a multi-controlled RZ gate with j controls (rather than $j-1$, because the overall control adds one additional control qubit), two controlled Hadamard gates, and $2(j-1)$ controlled-CNOT (Toffoli) gates.

The controlled phase gate contributes one single-qubit gate and zero CNOTs (it is a phase rotation on the control qubit). For each controlled-W_j, the multi-controlled RZ with j controls qubit decomposes into at most $16(j+1) - 40 = 16j - 24$ CNOTs for $j \geq 2$. The two controlled Hadamard gates contribute 2 CNOTs each (using the standard decomposition of controlled-H), and the $2(j-1)$ controlled-CNOTs become Toffoli gates, each requiring 6 CNOTs. Obtaining:

$$\begin{aligned} Q_{c\text{-}V_0}^{(\text{CNOT})} &= 2 + \sum_{j=2}^{n_x}(16j - 24) + 2n_x + 6\sum_{j=2}^{n_x} 2(j-1) \\ &= 2 + \left(16\frac{n_x(n_x+1)}{2} - 16 - 24(n_x - 1)\right) + 2n_x + 12\frac{(n_x-1)n_x}{2} \\ &= 2 + (8n_x^2 + 8n_x - 16 - 24n_x + 24) + 2n_x + 6(n_x^2 - n_x) \\ &= 2 + 8n_x^2 - 16n_x + 8 + 2n_x + 6n_x^2 - 6n_x \\ &= 14n_x^2 - 20n_x + 10 \end{aligned} \tag{6.3}$$

□

It is worth noting that the controlled operator has a larger gate count than the uncontrolled version, reflecting the additional resources required in order to condition the entire operation on a control qubit.

6.1.d The Full Heat-Equation Operator $V_{\text{heat}}(\tau)$

With the gate counts for V_0 and c-V_0 established, we now propagate these bounds to the full operator $V_{\text{heat}}(\tau)$. Recall from Equation (4.12) that V_{heat} is constructed using at most 2^{n_p-1} copies of $\widetilde{V}_0(-\tau)$ and $2^{n_p} - 1$ controlled copies of $\widetilde{V}_0(\tau)$, where $\widetilde{V}_0$ consists of d

copies of V_0 acting on independent spatial dimensions (one per dimension of the physical domain).

Lemma 9 (Gate count for V_{heat})**.** *For $n_x \geq 3$, the unitary $V_{\text{heat}}(\tau)$ of equation (4.12) can be implemented using $\mathcal{O}(dN_p n_x)$ single-qubit gates and at most*

$$Q_{V_{\text{heat}}} = d\,2^{n_p-1}\,Q_{V_0} + d\,(2^{n_p}-1)\,Q_{c\text{-}V_0} = \mathcal{O}(dN_p n_x^2) \tag{6.4}$$

CNOT gates.

Proof. By equation (4.13), the operator $\sum_{k=0}^{N_p-1} \tilde{V}_0^k(\tau) \otimes |k\rangle\langle k|$ is implemented via n_p controlled-power gates $c\text{-}\tilde{V}_0^{2^m}$, $m = 0, \ldots, n_p - 1$. The prefactor $\tilde{V}_0^{-N_p/2}(\tau)$ contributes one further application of $\tilde{V}_0$. In total there are at most 2^{n_p-1} uncontrolled $\tilde{V}_0$ gates and $2^{n_p} - 1$ controlled $\tilde{V}_0$ gates. Each $\tilde{V}_0$ consists of d copies of V_0, and each controlled $\tilde{V}_0$ consists of d copies of $c\text{-}V_0$. Multiplying through and using Lemma 7:

$$Q_{V_{\text{heat}}} = d\,2^{n_p-1} Q_{V_0} + d(2^{n_p}-1) Q_{c\text{-}V_0}.$$

Since $Q_{V_0}, Q_{c\text{-}V_0} = \mathcal{O}(n_x^2)$ and $2^{n_p-1} < N_p = 2^{n_p}$, we obtain $Q_{V_{\text{heat}}} = \mathcal{O}(dN_p n_x^2)$. The single-qubit count follows analogously: each V_0 requires $(2n_x + 1)$ single-qubit gates, and hence the decomposition requires at most 2^{n_p-1} uncontrolled $\tilde{V}_0$ applications, so the total single-qubit count is:

$$d \cdot 2^{n_p-1}(2n_x + 1),$$

that can be seen as $\mathcal{O}(dN_p n_x^2)$ total. □

6.1.e Gate Complexity of the Advection Equation Circuit

The complexity analysis for the advection equation follows a similar structural pattern to the one of the heat equation, though the presence of both Hermitian and skew-Hermitian components in the discretized operator introduces additional terms in the error bounds and requires separate treatment of the two component operators. Recall from Chapter 5 that the approximate operator $V_{\text{adv}}(\tau)$ is constructed from the operators V_1 and V_2, corresponding respectively to the Hermitian part H_1 and the skew-Hermitian part H_2 of the upwind-discretized advection operator.
If we consider the implementation of $V_{\text{adv}}(\tau)$, it involves a maximum of a $\tilde{V}_2(\tau)$ gate, 2^{n_p-1} $\tilde{V}_1(-\tau)$ gates and $\sum_{m=0}^{n_p-1} 2^m = 2^{n_p} - 1$ controlled $\tilde{V}_1(\tau)$ gates. Where, exactly as before, each $\tilde{V}_1$ ($\tilde{V}_2$) gate consists of d V_1 (V_2) gates. Moreover, V_1 and V_2 can be computed similarly to V_0 but they also have additional gates $U_1^{(1)}$ and $U_2^{(1)}$, each of which needs to be further broken down. As seen in eq (5.11) $U_1^{(1)}$ and $U_2^{(1)}$ are build almost in the same way and can be decompose into a multi-controlled RZ gate, plus the gates necessary to implement $B^{(1)}$ (resp. $B^{(2)}$), i.e. 2 Hadamards gates, 2 phase gates and $2(n_x - 1)$ X gates and $2(n_x - 1)$ CNOT gates. Meanwhile the controlled $U_1^{(1)}$ requires $2(n_x - 1)$ extra controlled X gates. Hence, the number of single-qubit gates is $d(4n_x + 2 + 2 + 2(n_x - 1)) + d2^{n_p-1}(2n_x + 2 + 2(n_x - 1) + 1) = \mathcal{O}(dN_p n_x)$.

While the total number of CNOT gates in the circuit implementation of V_{adv} is determined by

$$\mathcal{Q}_{V_{\text{adv}}} = d\mathcal{Q}_{V_2} + d2^{n_p-1}\mathcal{Q}_{V_1} + d(2^{n_p}-1)\mathcal{Q}_{c-V_1}, \tag{6.5}$$

with

$$\begin{aligned}\mathcal{Q}_{V_2} = \mathcal{Q}_{V_1} &= \mathcal{Q}_{V_0} + (16n_x - 40) + 2(n_x - 1)\\ &= 9n_x^2 - 15n_x - 6.\end{aligned} \tag{6.6}$$

$$\begin{aligned}\mathcal{Q}_{c-V_1} =& \mathcal{Q}_{c-V_0} + 16(n_x+1) - 40 + 2n_x + 16(n_x-1)\\ =& 14n_x^2 - 30.\end{aligned} \tag{6.7}$$

Then, V_{adv} can be implemented using $\mathcal{O}(dN_p n_x)$ single-qubit gates and at most

$$\begin{aligned}\mathcal{Q}_{V_{\text{adv}}} &= d\mathcal{Q}_{V_2} + d2^{n_p-1}\mathcal{Q}_{V_1} + d(2^{n_p}-1)\mathcal{Q}_{c-V_1}\\ &= \mathcal{O}(dN_p n_x^2),\end{aligned} \tag{6.8}$$

CNOT gates.

6.2 Total Gate Complexity of the Heat Equation

We now incorporate the Trotter step count and the measurement overhead into a single end-to-end gate-complexity bound.

Lemma 10 (One-step cost for $U_{\text{heat}}(T)$)**.** *Let the target Trotter precision be $\varepsilon' > 0$. The number of Trotter steps required to satisfy $\|U_{\text{heat}}(T) - V_{\text{heat}}^r(\tau)\| \le \varepsilon'$ is*

$$r \;\ge\; \frac{dN_p\gamma_0^2T^2(n_x-1)}{4\varepsilon'}. \tag{6.9}$$

Each step costs at most $\mathcal{O}(dN_p n_x^2)$ CNOT gates and $\mathcal{O}(dN_p n_x)$ single-qubit gates. Consequently, implementing $V_{\text{heat}}^r(\tau)$ costs

$$\mathcal{O}\left(d^2N_p^2n_x^3\,\frac{\gamma_0^2T^2}{\varepsilon'}\right) \text{ CNOT gates and } \mathcal{O}\left(d^2N_p^2n_x^2\,\frac{\gamma_0^2T^2}{\varepsilon'}\right) \text{ single-qubit gates.} \tag{6.10}$$

Proof. $V_{\text{heat}}^r(\tau)$ can be implemented using $O(r\mathcal{Q}_{V_{\text{heat}}}) = O(d^2n_x^3N_p^2\gamma_0^2T^2/\varepsilon)$ CNOT gates and $O(d^2n_x^2N_p^2\gamma_0^2T^2/\varepsilon)$ single-qubit gates. □

Theorem 1 (Gate complexity of the heat equation)**.** *Let $u : [0,T] \to L^2(\Omega)$ be the solution of the d-dimensional heat equation (4.1) with mesh size h and initial data $u_0 \in L^2(\Omega)$. The state $|u(T)\rangle$ can be prepared to within additive error ε (in the ℓ^2 sense on the grid) by the Schrödingerisation algorithm of using at most*

$$\widetilde{\mathcal{O}}\left(\frac{d^2T^2\|u_0\|^3}{\|u(T)\|^3h^4\varepsilon^3}\right) \tag{6.11}$$

single-qubit gates and CNOT gates in total.

Proof. The rigorous proof is well written in the article ([14], Theorem 1). □

6.3 Total Gate Complexity of the Advection Equation

Lemma 11 (One-step cost for $U_{\mathrm{adv}}(T)$)**.** *Set $C_{\mathbf{a}} := \sum_{\alpha=1}^{d} a_\alpha^2$ and let $\varepsilon' > 0$ be the target Trotter precision. The number of Trotter steps required for $\|U_{\mathrm{adv}}(T) - V_{\mathrm{adv}}^r(\tau)\| \le \varepsilon'$ is*

$$r \;\ge\; \frac{T^2 n_x C_{\mathbf{a}}}{4\varepsilon'}\big(N_p\gamma_2^2 + 2N_p\gamma_1\gamma_2 + 2\gamma_1^2\big). \tag{6.12}$$

Implementing $V_{\mathrm{adv}}^r(\tau)$ costs

$$\mathcal{O}\!\left(\frac{d\, n_x^3 T^2 N_p C_{\mathbf{a}}}{\varepsilon'}\big(N_p\gamma_2^2 + 2N_p\gamma_1\gamma_2 + 2\gamma_1^2\big)\right) \text{ CNOT gates.} \tag{6.13}$$

Proof. Identical structure to the proof of Lemma 10. $V_{\mathrm{adv}}^r(\tau)$ can be implemented using

$$\mathcal{O}(r\mathcal{Q}_{V_{\mathrm{adv}}}) = \mathcal{O}\left(dn_x^3 T^2 N_p(N_p\gamma_1^2 + 2N_p\gamma_1\gamma_2 + 2\gamma_2^2)\sum_{\alpha=1}^{d}\frac{a_\alpha^2}{\varepsilon}\right),$$

CNOT gates and $\mathcal{O}(dn_x^2 T^2 N_p(N_p\gamma_1^2 + 2N_p\gamma_1\gamma_2 + 2\gamma_2^2)\sum_{\alpha=1}^{d} a_\alpha^2/\varepsilon)$ single-qubit gates. □

Theorem 2 (Gate complexity of the advection equation)**.** *Let $u : [0,T] \to L^2(\mathbb{T}^d)$ be the solution of the d-dimensional advection equation (5.1) solved via the upwind scheme (5.3) with mesh size h. The state $|u(T)\rangle$ can be prepared to within additive error ε using at most*

$$\widetilde{\mathcal{O}}\left(\frac{d\, T^2 C_{\mathbf{a}} \|u_0\|^3}{\|u(T)\|^3 h^2 \varepsilon^3}\right) \tag{6.14}$$

single-qubit gates and CNOT gates in total.

Proof. Again, well written in the article ([14], Theorem 2). □

6.4 Classical vs. Quantum: A Rigorous Comparison

We now compare the complexities established in Theorems 1 and 2 with the cost of the best known classical finite-difference solvers for the same equations. Throughout, s denotes the sparsity of the relevant difference operator (i.e. the number of non-zero entries per row), and N_x^d is the total number of spatial degrees of freedom on the d-dimensional grid.

6.4.a Classical Complexity Baselines

Heat equation. Applying the forward Euler scheme to (4.4), the global temporal discretization error is $\mathcal{O}(\tau)$, while stability requires the parabolic CFL condition $\tau = \mathcal{O}(h^2)$. Hence the number of time steps needed to reach time T is

$$N_t = \mathcal{O}\left(\max\left\{\frac{T}{\varepsilon}, \frac{T}{h^2}\right\}\right),$$

where the first term enforces the desired temporal accuracy and the second the CFL stability constraint. Each step requires one sparse matrix–vector multiplication with the finite-difference Laplacian, whose stencil has sparsity $s = s(D_D^{\Delta}) = 2d+1$, costing $\mathcal{O}(sN_x^d)$ arithmetic operations. The resulting classical complexity is therefore

$$C_{\mathrm{cl}}^{\mathrm{heat}} = \mathcal{O}\left(sN_x^d \max\left\{ \frac{T}{\varepsilon}, \frac{T}{h^2} \right\} \right) = \mathcal{O}\left(s\, h^{-d} \max\left\{ \frac{T}{\varepsilon}, \frac{T}{h^2} \right\} \right). \tag{6.15}$$

For the second-order spatial discretization, balancing the spatial error $\mathcal{O}(h^2)$ with the temporal error $\mathcal{O}(\tau) = \mathcal{O}(\varepsilon)$ gives $h = \mathcal{O}(\sqrt{\varepsilon})$, yielding

$$C_{\mathrm{cl}}^{\mathrm{heat}} = \mathcal{O}\left(s\, T\, \varepsilon^{-(d/2+1)} \right).$$

Advection equation. For the first-order upwind discretization, both the spatial and temporal discretization errors are first order, $\mathcal{O}(h)$ and $\mathcal{O}(\tau)$, respectively. The hyperbolic CFL condition requires $\tau = \mathcal{O}(h)$, so the number of time steps is

$$N_t = \mathcal{O}\left(\max\left\{ \frac{T}{\varepsilon}, \frac{T}{h} \right\} \right).$$

Each step costs $\mathcal{O}(sN_x^d)$ operations with stencil sparsity $s = d + 1$, giving

$$C_{\mathrm{cl}}^{\mathrm{adv}} = \mathcal{O}\left(sN_x^d \max\left\{ \frac{T}{\varepsilon}, \frac{T}{h} \right\} \right) = \mathcal{O}\left(s\, h^{-d} \max\left\{ \frac{T}{\varepsilon}, \frac{T}{h} \right\} \right). \tag{6.16}$$

Balancing the first-order spatial and temporal errors gives $h = \mathcal{O}(\varepsilon)$, for which

$$C_{\mathrm{cl}}^{\mathrm{adv}} = \mathcal{O}\left(s\, T\, \varepsilon^{-(d+1)} \right).$$

6.4.b Quantum Complexity: Restated in Comparable Terms

Setting $h = \mathcal{O}(\varepsilon)$ and $\|u_0\|/\|u(T)\| = \mathcal{O}(1)$ in Theorems 1 and 2:

$$C_{\mathrm{qu}}^{\mathrm{heat}} = \widetilde{\mathcal{O}}\left(\frac{d^2 T^2}{\varepsilon^3} \right), \tag{6.17}$$

$$C_{\mathrm{qu}}^{\mathrm{adv}} = \widetilde{\mathcal{O}}\left(\frac{d\, T^2 C_{\mathbf{a}}}{\varepsilon^3} \right). \tag{6.18}$$

6.4.c Comparison and Dimension Thresholds

Table 6.1: Comparison of classical and quantum gate/operation complexities for the heat and advection equations. For the heat equation we balance the second-order spatial error with the first-order temporal error by taking $h = \mathcal{O}(\sqrt{\varepsilon})$; for the advection equation we take $h = \mathcal{O}(\varepsilon)$. Logarithmic factors are suppressed, and $\|u_0\|/\|u(T)\| = \mathcal{O}(1)$. Here s denotes the sparsity of the corresponding finite-difference operator.

Quantity	**Heat equation**	**Advection equation**
Classical state space	$\mathcal{O}(N_x^d) = \mathcal{O}(\varepsilon^{-d/2})$	$\mathcal{O}(N_x^d) = \mathcal{O}(\varepsilon^{-d})$
Quantum state space	$dn_x + n_p = \mathcal{O}(d\log(1/\varepsilon))$ qubits	$dn_x + n_p = \mathcal{O}(d\log(1/\varepsilon))$ qubits
Classical cost / step	$\mathcal{O}(s\,\varepsilon^{-d/2})$	$\mathcal{O}(s\,\varepsilon^{-d})$
Quantum cost / step	$\tilde{\mathcal{O}}\big(d\,\varepsilon^{-1}\log^2(1/\varepsilon)\big)$	$\tilde{\mathcal{O}}\big(d\,\varepsilon^{-1}\log^2(1/\varepsilon)\big)$
Classical total operations	$\mathcal{O}(s\,T\,\varepsilon^{-(d/2+1)})$	$\mathcal{O}(s\,T\,\varepsilon^{-(d+1)})$
Quantum total gates	$\tilde{\mathcal{O}}(d^2T\,\varepsilon^{-3})$	$\tilde{\mathcal{O}}(dT\,\varepsilon^{-3})$
Asymptotic advantage threshold	$\boldsymbol{d > 4}$	$\boldsymbol{d > 2}$

Remark: 11 (Exponential state-space compression)**.** *Even before considering time complexity, the quantum algorithm provides an exponential reduction in the memory required to represent the state: a classical solver needs $\mathcal{O}(N_x^d) = \mathcal{O}(2^{dn_x})$ real numbers, whereas the quantum state is stored in $dn_x + n_p$ qubits. This exponential compression is unconditional and does not depend on the dimension threshold. However, it is contingent on amplitude encoding, which requires an efficient state-preparation subroutine (assumed here as a black box).*

6.5 Oracle-Free Complexity and Comparison with Prior Work

Definition 7 (Oracle-free circuit)**.** *A quantum circuit for a PDE solver is* oracle-free *if every gate in the circuit is explicitly specified, and therefore can be broken down as a sequence of single-qubit and CNOT gates, with no unresolved subroutines.*

The circuits of Chapters 4 and 5 satisfy Definition 7: every step, from the W_j building blocks, through the controlled-power assembly to the QFT, is expressed in terms of elementary gates with explicit CNOT counts. For instance, this stands in contrast to the HHL-based approach of Cao et al. [4] and the oracle-dependent methods of Childs, Liu and Ostrander [6].

Comparison with Jin, Liu and Yu [12, 13]. The theoretical framework of Schrödingerisation was established in [12, 13] at the level of continuous operators and semi-discrete

ODE systems. The present work and [14] constitute the first *fully explicit circuit implementation* of this framework, extending the analysis to concrete gate sequences and oracle-free CNOT counts.

Complexity of the QFT. The (inverse) quantum Fourier transform over n_p qubits is implemented with $\mathcal{O}(n_p^2)$ CNOT gates [20]. Since $n_p = \mathcal{O}(\log(1/\varepsilon))$, this contributes only $\mathcal{O}(\log^2(1/\varepsilon))$ gates, which is negligible compared to the dominant cost of $V_{\mathrm{heat}}^r(\tau)$ or $V_{\mathrm{adv}}^r(\tau)$.

Remark: 12 (Hardware requirements)**.** *The total qubit count of the algorithm is* $n = dn_x + n_p = \mathcal{O}(d\log(L/h) + \log(1/\varepsilon))$*, which is logarithmic in both the spatial resolution and the target precision. For a three-dimensional problem* ($d = 3$) *with* $h = 0.01$ *and* $\varepsilon = 0.01$*, one needs approximately* $3 \times 7 + 7 = 28$ *logical qubits. While this is within the reach of near-term devices in terms of qubit count, the circuit depth, which scales as* $r \cdot \mathrm{depth}(V_{\mathrm{heat}}) = \mathcal{O}(rn_xn_p)$*, and the fidelity requirements necessitate full fault-tolerant quantum computation for any practically meaningful problem size. The present results are therefore most accurately can be described as characterising the asymptotic regime of quantum advantage rather than the near-term regime.*

7 Visualising Quantum Simulation Results: State Vectors, Samplers, and Estimators

We will now provide a detailed account of how the numerical results produced by the quantum Schrödingerization algorithm of our article [14] are extracted from the quantum simulation and subsequently visualized. This pipeline is crucial to understanding what we mean by "quantum simulation", and what results we might achieve using real quantum hardware in the future; such understanding requires distinguishing three conceptually different objects that Qiskit, the SDK we used to simulate quantum hardware, offers: the *statevector*, the *Sampler*, and the *Estimator*. Each object encodes a different aspect of the quantum state, and each produces a different type of data that is plotted for different physical purposes. After the discussion about the code from the paper in detail, we present a pedagogical example that isolates the plotting mechanics from the time-evolution machinery, making the differences between the three approaches visually transparent. We suggest to the readers that are already familiar with these tools to proceed to the next Chapter 8, where the results of the various numerical experiments are discussed.

7.1 Background: The Three Modes of Accessing a Quantum State

Recall that a pure quantum state on n qubits is a unit vector

$$|\psi\rangle = \sum_{k=0}^{2^n-1} \alpha_k |k\rangle \qquad \text{s.t.} \quad \alpha_k \in \mathbb{C}, \quad \sum_{k=0}^{2^n-1} |\alpha_k|^2 = 1, \tag{7.1}$$

where $\{|k\rangle\}$ conventionally denotes the computational basis.
In a classical simulation running on a computer (as opposed to real quantum hardware), the full amplitude vector $(\alpha_0, \ldots, \alpha_{2^n-1}) \in \mathbb{C}^{2^n}$ is available in memory. On real quantum hardware, by contrast, as we measure we destroy the superposition information; the informations we can actually access are only partial and are constrained by the postulate known as the Born Rule which tells us that while measuring we can only retrieve the probability $|\alpha_k|^2$ that the state $|\psi\rangle$ yields outcome k.

Qiskit models these two regimes through three primitives:

Statevector / `Statevector.data`. The object `Statevector(qc).data` returns the full com-

plex amplitude vector $(\alpha_0, \ldots, \alpha_{2^n-1})$ as a `numpy` array. This is only possible in a classical simulation. Indeed measuring a real qubit "collapses" the state so regardless of the quantum hardware used one cannot retrieve the amplitudes directly.
It gives complete, noiseless information about the quantum state, at exponential memory cost in the number of qubits.

`StatevectorSampler`. The Sampler mimics what happens on real quantum hardware: it applies the Born rule repeatedly for a prescribed number of shots (identical trials). Notice that Qiskit supports multiple simulation methods and configurable options for each simulation method; the study of the various different types of simulations is beyond the scope of this work, if the reader would like to delve further into this aspect, please refer to QiskitAer documentation.
Each shot collapses the state to a basis vector $|k\rangle$ with probability $|\alpha_k|^2$, and the output is a histogram (bit-string counts). From these counts one can estimate probabilities, but *not* the individual amplitudes α_k.

`StatevectorEstimator`. The Estimator computes the expectation value of a Hermitian observable O,

$$\langle O \rangle = \langle \psi | \, O \, | \psi \rangle \,, \tag{7.2}$$

given a quantum circuit and a Pauli operator. On a classical simulator this is computed exactly (or with controlled precision); on real hardware it requires many shots and statistical averaging.

The article's authors use all three objects for different purposes, which we will now carefully explain.

7.2 The Schrödingerization Algorithm and Its Output

The algorithm simulates the equation $\partial_t u = Au$ (where A is generally non-Hermitian) by embedding $u(t, \cdot)$ into a larger Hamiltonian system. We will delve deeper into how this happens in the next chapters. At this point the reader just needs to know that the quantum state at time t lives in a joint Hilbert space $\mathscr{H}_x \otimes \mathscr{H}_p$, where $\mathscr{H}_x$ represents the spatial degree of freedom (encoded on n_x qubits) and $\mathscr{H}_p$ is an auxiliary momentum register (encoded on $n_p =$ `na` qubits).

The physical solution $u(t, \cdot)$ is then recovered projecting the joint state onto the subspace $\{p = 0\}$.
In the discrete setting this corresponds to selecting a specific row of the reshaped amplitude tensor.

7.2.a Recovering the Solution via the Statevector (`plot_solution`)

The function `schro` in the notebook performs the following extraction at the end of every time step and at the final time T:

```
                    label=lst:statevector]
ut = Statevector(qc).data      # complex array of length 2^(nx+na)
qc.clear()
qc.initialize(ut / np.linalg.norm(ut), qc.qubits)# reset for next
    step
```

At the final time, after the inverse QFT has been applied we have the following line of code that, as previously mentioned, project our solution onto a subspace:

```
u = np.real(Statevector(qc)).reshape(2**na, 2**nx)
u = u[2**(na-1)] * norm    # row index 2^(na-1) corresponds to \eta
    =0
```

Here the amplitude vector of length $2^{n_x+n_p}$ is reshaped into a $2^{n_p} \times 2^{n_x}$ matrix; row 2^{n_p-1} corresponds to the zero-frequency (i.e. $p = 0$) slice.
The multiplication by `norm` is needed in order to reverse the normalization applied when the state was initialized.

The resulting array `u` is a real vector of length 2^{n_x} approximating the PDE solution $u(T, x_j)$ at the spatial grid points x_j.

Remark: 13. *The imaginary part is discarded with `np.real()`. For the heat equation the exact solution is real, so this is mathematically justified; any residual imaginary part is purely numerical noise from the circuit simulation.*

7.2.b What `plot_solution` Plots

The function `plot_solution` (Figures 8.1, 8.3) calls `run` twice for each value of $n_p \in \{3, 5, 7\}$ (one coarse time step Δt_1 and one fine step Δt_2) and then plots five curves on the same axes:

Curve	Source	Colour / marker
Exact solution	Analytical formula	Red, solid
Matrix exponential	$e^{AT}u_0$ via `scipy`	Blue, dashed
Classical Schrödingerization	Direct matrix simulation	Black, dashed + squares
Proposed (coarse Δt)	Quantum circuit + statevector	Diamond, default colour
Proposed (fine Δt)	Quantum circuit + statevector	Diamond, default colour

The horizontal axis is the spatial variable x, and the vertical axis is the function value $u(T, x)$. All four numerical methods operate on the same spatial grid, so the curves are comparable.

The key comparison being made is:

1. **Accuracy vs. exact solution**: how close is each numerical method to the red curve?
2. **Time-step sensitivity**: comparing the two quantum curves (large and small Δt) reveals the time-discretization error of the proposed Schrödingerization scheme.

3. **Effect of n_p**: the three subplots (one per column) show how increasing the number of ancilla qubits n_p improves the approximation of the p-space integral and hence the accuracy of the method.

All data in this plot come exclusively from `Statevector.data`. No measurement simulation or observable expectation is involved.

7.3 Measuring the L^2 Norm via Sampler and Estimator (`plot_norm`)

The function `plot_norm` tracks the squared L^2 norm $\|u(t)\|^2$ over time $t \in [0, T]$; the squared norm (total "energy") of the solution u to the heat equation at each time t (x-axis). Such a quantity for the heat equation decays exponentially, while for the advection equation it is (ideally) conserved. Monitoring it serves as a diagnostic of how faithfully the quantum circuit preserves the dynamics. Moreover it is an example for which type of information we could actually get on real quantum hardware.

7.3.a Why Two Quantum Estimations?

The state after applying the inverse QFT is supported on all momentum modes. The physical solution corresponds only to $p = 0$, but measuring $\|u(t)\|^2$ from the quantum state requires knowledge about what fraction of the total norm sits at $p = 0$. The notebook uses *two* independent quantum measurements:

Measurement 1 — Estimator (expectation value, $p = 0$ slice).
The observable

$$O = \underbrace{I^{\otimes n_x}}_{x} \otimes \underbrace{|1\rangle\langle 1|_{p_0}}_{\text{selects } p_0\text{-qubit}=1} \otimes \underbrace{|0\rangle\langle 0|^{\otimes (n_p-1)}}_{\text{rest of } p} \tag{7.3}$$

is built in the code as a tensor product of Pauli operators. In order to understand how it is built observe that we can construct both the zero operator and the one operator as:

$$|0\rangle\langle 0| = \frac{1}{2}(I + Z);$$

$$|1\rangle\langle 1| = \frac{1}{2}(I - Z).$$

```
zero_op = SparsePauliOp(['I', 'Z'], coeffs=[0.5,  0.5])  # |0><0|
one_op  = SparsePauliOp(['I', 'Z'], coeffs=[0.5, -0.5])  # |1><1|

observable = one_op                    #acts on p_0 qubit
for i in range(na - 1):
    observable = observable ^ zero_op #remaining p qubits in |0>
observable = observable ^ obs          #identity over x qubits

###when the function shro is called obs is defined as:###
            ###obs = Pauli('I' * nx)###

```

```
circuit = qc.copy()
circuit.x(ra[-1])
circuit.append(iqft, ra)
pub = (circuit, observable)
job = estimator.run([pub], precision=np.sqrt(var[i])/100)
values[0, i] = job.result()[0].data.evs * norm**2
```

The Estimator then computes

$$\langle O \rangle = \langle \Psi(t) | \, O \, | \Psi(t) \rangle \approx \frac{\|u(t)\|^2}{\text{norm}^2}, \tag{7.4}$$

and the code rescales by `norm`^{2} to recover $\|u(t)\|^2$. This is stored in `values[0, i]`.

Remark: 14. *The precision of the Estimator is set to* $\sqrt{\text{Var}}/100$*, where* $\text{Var} = E - E^2$ *and* E *is the classically precomputed expectation. This is a* target standard deviation *for the estimate; the Estimator internally adjusts the number of shots to achieve it.*

Measurement 2 — Sampler (probability, $p \geq 0$).
These lines represent the most standard approach to measurement on actual quantum hardware:

```
alpha = ClassicalRegister(1, name='alpha')
circuit.add_register(alpha)
circuit.measure(ra[-1], alpha)
job = sampler.run([circuit])
prob_pge0 = job.result()[0].data.alpha.get_counts()['1'] / 10000
values[1, i] = prob_pge0 * norm**2 / norm1**2
```

We can retrieve the energy obtained from the circuits by **measuring** the last qubit in register p (qubit `ra[-1]`) to obtain the probability $P(1)$ of it being in state $|1\rangle$. In order to do that we add a classical bit `alpha` to our circuit and we measure the last qubit overwriting such measurement onto `alpha`. Then the sampler run such circuit 10,000 times. The `get_counts()['1']` gives us the number of times such qubit was measured as 1, i.e. the number of shots that landed in the $pk \geq 0$ half of the p-register. Dividing by 10000 gives the empirical probability:

$$P(1) \approx \frac{\text{counts}['1']}{10000}.$$

Finally we get a weighted estimate of $\|u(T)\|^2$ as:

$$\|u(T)\|^2 \approx \frac{P(1)\|u(0)\|^2\|p\|^2}{\|p_{\geq 0}\|^2}. \tag{7.5}$$

Remark: 15. *The scatter we see in the dots' plotting is due to shot noise, the circuit was run 10,000 times and probabilities are estimated from such samples; in other words the spread is statistical, not structural, and it should tighten with more shots.*

7.3.b What `plot_norm` Plots

The function plots five time series over $t \in [0, T]$:

Curve	Source	Colour
Exact $\|u\|^2$	Analytical	Red
Matrix exponential $\|u\|^2$	`scipy`	Blue, dashed
Classical Schrödingerisation $\|u\|^2$	Direct matrix	Black
Proposed ($p = 0$)	`Estimator`	Medium sea green, dotted
Proposed ($p \geq 0$)	`Sampler`	Orange, dotted

The horizontal axis is time t, and the vertical axis is $\|u(t)\|^2$. The two quantum curves together demonstrate that:

- The Estimator ($p = 0$) recovers $\|u(t)\|^2$ by measuring the expectation of a projection operator.
- The Sampler ($p \geq 0$) recovers $\|u(t)\|^2$ by measuring raw shot-based probabilities.

Both should converge to the exact curve as n_p increases and the number of shots increases.

Remark: 16. *We have not discussed about the Classical Schrödingerisation plot. With the term Classical Schrödingerisation the autors refer to the direct classical matrix simulation of the same Schrödingerisation scheme, without any quantum circuit. It is based on the mathematical ground truth on which the quantum circuit is supposed to compute, before any quantum-level approximations are introduced. The difference between this calssical curve and the quantum diamond curves in* `plot_solution` *is the Trotterisation error due to the fact that the quantum circuit needs to approximate $e^{iH\Delta t}$ with a product of simpler gates rather than computing the full matrix exponential at once.*

7.4 Summary Scheme

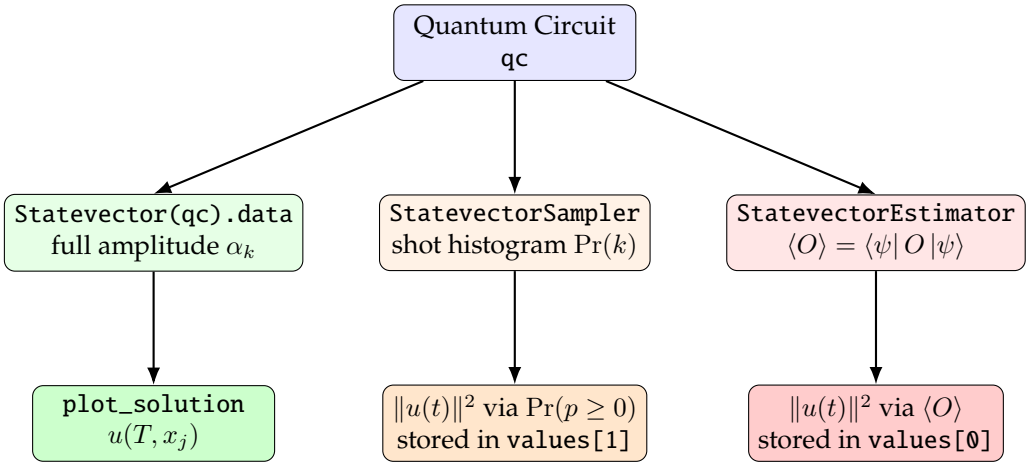


Figure 7.1: Information flow from the quantum circuit to the two plotting functions. The statevector gives direct access to amplitudes; the Estimator gives expectation values; the Sampler gives measurement-count histograms.

7.5 A Self-Contained Pedagogical Example

In order to isolate the plotting mechanics from the complexity of the Schrödingerisation scheme, we now present a minimal example.

7.5.a Setup

Consider a discretize function $f : \{0, \dots, 2^n - 1\} \to \mathbb{R}$, encode it as a quantum state, and then recover and plot it using each of the three methods in turn. Let $n = 3$ (In such way the Hilbert space associated to the quantum circuit has dimension $N = 2^3 = 8$), and define f as

$$f_k = \sin\left(\frac{2\pi k}{N}\right), \qquad k = 0, 1, \dots, N-1. \tag{7.6}$$

The corresponding normalised quantum state is

$$|\psi\rangle = \frac{1}{\|f\|} \sum_{k=0}^{N-1} f_k \, |k\rangle \,, \qquad \|f\| = \left(\sum_{k=0}^{N-1} f_k^2\right)^{1/2}. \tag{7.7}$$

We initialise a 3-qubit circuit with this state using `qc.initialize`.

```
import numpy as np
from qiskit import QuantumCircuit
from qiskit.quantum_info import Statevector

n = 3
N = 2**n
k = np.arange(N)
f = np.sin(2 * np.pi * k / N)       # the function to encode
norm = np.linalg.norm(f)

# Encode f as a quantum state
qc = QuantumCircuit(n)
qc.initialize(f / norm, range(n))
```

Remark: 17. *The vector f has both positive and negative entries. The Born rule gives*

$$\Pr(\text{outcome } k) = |f_k|^2 / \|f\|^2,$$

so the Sampler cannot distinguish f_k from $-f_k$ which we noticed to be a fundamental limitation of measurement-based methods.

7.5.b Method 1: Full Recovery via Statevector

```
# Recover the full amplitude vector
sv = Statevector(qc).data           # complex array of length N
f_recovered = np.real(sv) * norm    # rescale to physical units

```

```
# Plot
import matplotlib.pyplot as plt
plt.figure(figsize=(7, 4))
plt.plot(k, f,'k-o', label=r'Original $f_k$')
plt.plot(k, f_recovered, 'r--s', label='Recovered (Statevector)')
plt.xlabel(r'$k$')
plt.ylabel(r'$f_k$')
plt.title('Function recovery via Statevector')
plt.legend()
plt.tight_layout()
plt.show()
```

The statevector gives *exact recovery* (up to floating-point precision): $f_k^{\text{rec}} = f_k$ for all k. This is because `Statevector.data` returns the amplitude vector directly, without any stochastic element.

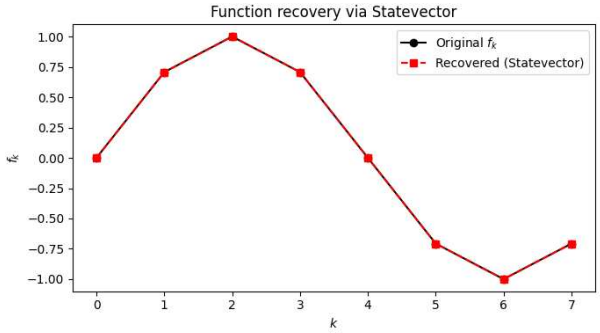


Figure 7.2: Output

Remark: 18. *In the Schrödingerisation notebook, this method is used in exactly the same way to recover $u(T, x_j)$ at the end of the time evolution. The only additional step is the reshaping and row-selection that implements the projection to $p = 0$.*

7.5.c Method 2: Probability Estimation via the Sampler

The Sampler can only access the *squared* amplitudes. We add a measurement gate and run the circuit for a number of shots.

```
                    label=lst:sampler-example]
from qiskit.primitives import StatevectorSampler
from qiskit import ClassicalRegister

# Add measurement
cr = ClassicalRegister(n, name='c')
qc_meas = qc.copy()
qc_meas.add_register(cr)
qc_meas.measure(range(n), cr)

# Run sampler with varying shot counts
```

```
sampler = StatevectorSampler()
shots_list = [100, 1000, 10000]
fig, axes = plt.subplots(1, 3, figsize=(15, 4), sharey=True)

for ax, shots in zip(axes, shots_list):
    sampler_shots = StatevectorSampler(default_shots=shots)
    job = sampler_shots.run([qc_meas])
    counts = job.result()[0].data.c.get_counts()

    # Convert counts to probability estimates
    probs_est = np.array([counts.get(format(k, f'0{n}b'), 0) for k
    in range(N)]) / shots
    # probs_est ~ |f_k|^2 / norm^2, so rescale to get |f_k|
    f_abs_est = np.sqrt(probs_est) * norm

    ax.bar(k, f**2 / norm**2, alpha=0.5, label=r'True $|f_k|^2/\|f
    \|^2$')
    ax.bar(k, probs_est,      alpha=0.5, label='Sampler estimate')
    ax.set_xlabel(r'$k$')
    ax.set_title(f'{shots} shots')
    if ax == axes[0]:
        ax.set_ylabel('Probability')
    ax.legend(fontsize=8)

fig.suptitle('Probability estimation via Sampler (different shot
    counts)', y=1.02)
plt.tight_layout()
plt.show()
```

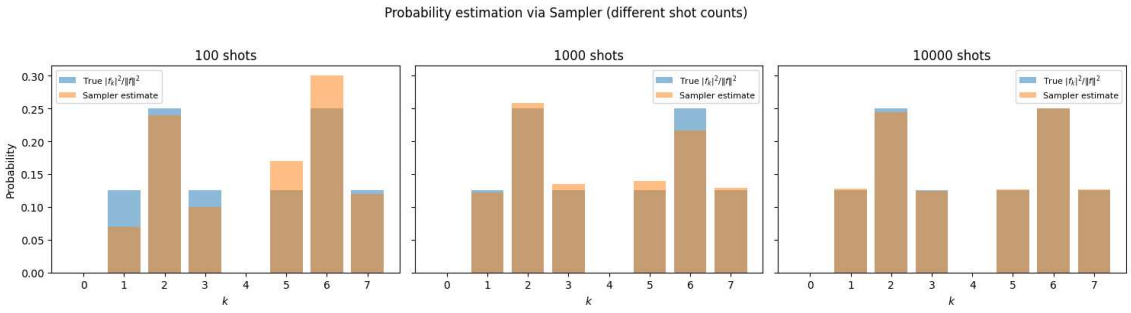


Figure 7.3: Output

Several observations follow from this experiment:

1. The Sampler output is a *histogram*, not a function. It gives estimates of $|f_k|^2/\|f\|^2$, not of f_k itself.

2. With few shots (e.g. 100), the histogram is noisy; with many shots (e.g. 10,000) it converges to the true probability distribution.

3. Since $|f_k|^2 = |-f_k|^2$, the Sampler cannot distinguish a positive-valued point from a negative-valued one.

In the Schrödingerization notebook, the Sampler is used precisely for a task where sign information is not needed: estimating the *norm* $\|u(t)\|^2$, which is manifestly non-negative. The probability of measuring the qubit `ra[-1]` in state $|1\rangle$ is proportional to the total weight of the state in the $p \geq 0$ half of momentum space, from which $\|u(t)\|^2$ can be recovered up to a known normalization constant.

7.5.d Method 3: Expectation Value via the Estimator

The Estimator computes $\langle O \rangle = \langle \psi | O | \psi \rangle$ for a specified Pauli observable O. To illustrate this, we compute the expectation values of Z_j on each qubit:

$$\langle Z_j \rangle = \sum_{k=0}^{N-1} |\alpha_k|^2 \cdot (-1)^{b_j(k)}, \tag{7.8}$$

where $b_j(k)$ is the j-th bit of k in binary.

```
                    label=lst:estimator-example]
from qiskit.primitives import StatevectorEstimator
from qiskit.quantum_info import SparsePauliOp

estimator = StatevectorEstimator()

# Build observables: Z on each qubit, I on the rest
pauli_strings = []
for j in range(n):
    s = ['I'] * n
    s[j] = 'Z'
    pauli_strings.append(''.join(s))

observables = [SparsePauliOp(p) for p in pauli_strings]

# Compute expectation values
evs = []
for obs in observables:
    pub = (qc, obs)
    job = estimator.run([pub])
    evs.append(job.result()[0].data.evs)

# Plot
fig, axes = plt.subplots(1, 2, figsize=(12, 4))

# Left: the original function
axes[0].plot(k, f, 'k-o')
axes[0].set_xlabel(r'$k$')
axes[0].set_ylabel(r'$f_k$')
```

```
axes[0].set_title('Original function $f_k = \sin(2\pi k/N)$')

# Right: the qubit-wise expectation values <Z_j>
axes[1].bar(range(n), evs, color='steelblue')
axes[1].set_xlabel('Qubit index $j$')
axes[1].set_ylabel(r'$\langle Z_j \rangle$')
axes[1].set_title('Estimator: per-qubit Pauli-$Z$ expectations')
axes[1].set_xticks(range(n))

plt.tight_layout()
plt.show()
```

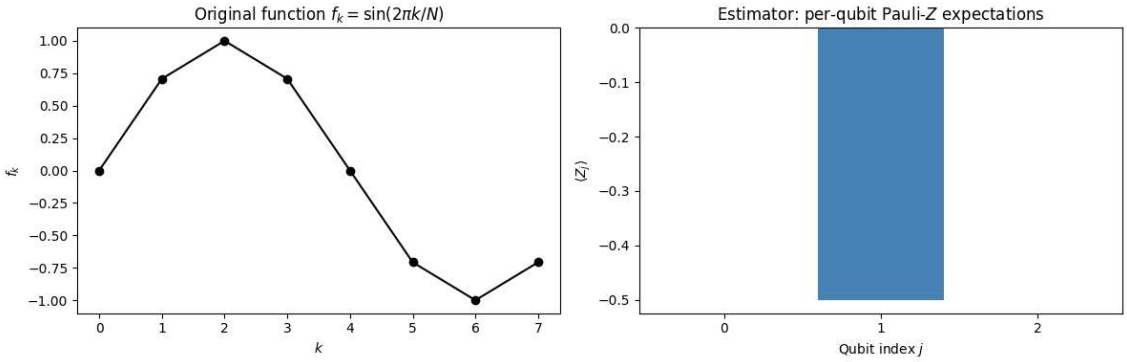


Figure 7.4: Output

The right panel shows three numbers, one per qubit, which are weighted averages over the distribution $|f_k|^2/\|f\|^2$, not the function f_k itself. The Estimator is powerful not for reconstructing a function point-wise, but for computing *aggregate scalar quantities* (such as: norms, energies, or overlaps) that are expressible as expectation values of known observables.

In the Schrödingerization notebook, the Estimator computes a projection observable that selects the $p = 0$ slice, effectively computing $\|u(t)\|^2$.

7.6 Conclusion

The three Qiskit primitives — `Statevector.data`, `StatevectorSampler`, and `StatevectorEstimator` — answer fundamentally different questions about a quantum state and are used in the notebook for three distinct purposes:

1. **Statevector** is used wherever the full classical simulation is available: to extract the PDE solution $u(T, x_j)$ at the final time and to reset the circuit state at each time step (avoiding circuit depth blow-up).

2. **Sampler** is used to estimate the probability of measuring the $p \geq 0$ half-space, yielding an estimate of $\|u(t)\|^2$ via raw shot counts — the primitive that most faithfully models real quantum hardware.

3. **Estimator** is used to compute the expectation value of a structured projection observable, giving a statistically cleaner (lower variance) estimate of $\|u(t)\|^2$ via the $p = 0$ slice.

Understanding this distinction is essential not only for reading the plots produced by the notebook, but also for any future work that seeks to run the algorithm to real quantum hardware, where the statevector is unavailable and only Sampler- and Estimator-based strategies remain viable.

8 Numerical Experiments

The theoretical developments presented in the preceding chapters get their full significance only when validated through numerical implementation. This chapter presents a comprehensive suite of numerical experiments designed to verify the correctness of the quantum circuits proposed in [14], to quantify their approximation accuracy, and to illustrate the practical behavior of the Schrödingerization method on a classical quantum simulator.

As mentioned, all experiments were performed using the Qiskit package [11]. The numerical results are compared against two independent reference solutions: a direct classical implementation of the Schrödingerization method using matrix algebra (via SciPy [23]), and the exact matrix exponential e^{AT} applied to the initial condition. These three-way comparisons enable a precise attribution of errors to their respective sources: spatial discretization error, momentum-truncation error, and Trotter-splitting error.

The reader is assumed to be familiar with the three Qiskit primitives (Statevector, Sampler, and Estimator) which are discussed in detail in Chapter 7. The present chapter focuses on the experimental design, the numerical results, and their interpretation. We begin by describing the problem setup and implementation parameters, then present the results for the heat equation and the advection equation, and conclude with a detailed discussion of the error decomposition and convergence properties.

8.1 Problem Setup and Implementation Parameters

Consistently with the circuits we have developed in the Chapters 4 and 5 we consider two model problems: the one-dimensional heat equation with Dirichlet boundary conditions, and the one-dimensional advection equation with periodic boundary conditions. These problems, while mathematically simple, capture the essential features of dissipative and hyperbolic PDEs, respectively, and provide the ideal baseline test cases for the Schrödingerization framework. The one-dimensional setting is chosen to enable detailed visualization and precise error quantification; the extension to higher dimensions follows the tensor product structure.

8.1.a Heat Equation: Parameters and Discretization

The heat equation on the spatial domain $\Omega = [0, L]$ is given by

$$\begin{cases} \partial_t u(t,x) = a\partial_{xx} u(t,x), & x \in [0,L],\ t > 0, \\ u(0,x) = f(x), & x \in [0,L], \\ u(t,0) = u(t,L) = 0, & t > 0, \end{cases} \tag{8.1}$$

where the diffusion coefficient and initial condition are chosen as

$$a = \frac{L}{\pi^2}, \qquad f(x) = \sin\Big(\frac{\pi x}{L}\Big). \tag{8.2}$$

With this particular choice of parameters, the exact analytical solution is:

$$u(t,x) = e^{-t/L} \sin\Big(\frac{\pi x}{L}\Big), \tag{8.3}$$

which provides a ground truth against which all numerical approximations are measured. The decay rate $1/L$ ensures that the solution remains non-negligible at the final simulation time $T = 5$, while still exhibiting the characteristic dissipative behavior of the heat equation.

For the spatial discretization, keeping the formulation consistent with what we have already developed , we employ the standard second-order central difference scheme on a uniform grid with $N_x = 2^{n_x}$ interior points. In the results reported here, we set $n_x = 4$, corresponding to $N_x = 16$ interior grid points and a mesh size $h = L/(N_x + 1) = L/17$.

The Schrödingerization parameters are chosen as follows to systematically explore the convergence properties of the method. The domain of p is chosen to be $[-4\pi, 4\pi]$; Observe that with the values chosen large the truncated tail of the warped phase function $e^{-|p|}$ is negligible ($e^{-4\pi} \approx 10^{-6}$). To discretize such domain we have used different grids, in particular the grids use $N_p = 2^{n_p}$ points with $n_p \in \{3, 5, 7\}$, corresponding to $N_p \in \{8, 32, 128\}$. These three values of n_p allow us to study the convergence of the method with respect to the "resolution" of p. We also have implemented the method on two different time steps: in one case we have used $\tau = 0.005$, requiring $r = T/\tau = 1000$ Trotter steps; in the other case we have used a coarser time step $\tau = 0.5$ ($r = 10$ steps) in order isolate the effect of Trotter-error accumulation.

The discrete initial state $|u(0)\rangle$ is prepared by encoding the normalized initial condition vector into the n_x-qubit spatial register, while the momentum register is initialized to the normalized warped phase vector $\mathbf{p}$ with components $p_k = e^{-|p_k|}$ for $k = 0, \ldots, N_p - 1$, where $p_k = -4\pi + 2\pi 4k/N_p$.

8.1.b Advection Equation: Parameters and Discretisation

The advection equation on the periodic domain $\Omega = [0, L]$ is given by

$$\begin{cases} \partial_t u(t,x) = a\partial_x u(t,x), & x \in [0,L],\ t > 0, \\ u(0,x) = f(x), & x \in [0,L], \end{cases} \tag{8.4}$$

with periodic boundary conditions $u(t,0) = u(t,L)$. We have set $a = 1$ and chosen a discontinuous initial condition in the form of a square wave:

$$f(x) = \begin{cases} 0, & x \in \left[kL, kL + \frac{L}{2}\right), \\ 1, & x \in \left[kL + \frac{L}{2}, (k+1)L\right), \end{cases} \qquad k \in \mathbb{Z}. \tag{8.5}$$

Choosing a discontinuous initial condition allows us to test the ability of the numerical method to handle sharp gradients without introducing spurious oscillations. We recall that in this case the exact solution is the traveling wave

$$u(t,x) = f(x + at). \tag{8.6}$$

For the spatial discretization, we employ the first-order upwind scheme rather than central differences, accordingly with what we have discussed in Subsection 5.1.b. The upwind matrix A is given by Equation (5.4), as before with $N_x = 2^{n_x} = 16$ grid points and mesh size $h = L/N_x = L/16$. Also the Schrödingerization parameters used for the advection equation are identical to those for the heat equation: $R = 4$, $n_p \in \{3, 5, 7\}$, and $\tau \in \{0.5, 0.005\}$, but with final time $T = 3$, this implies having $r \in \{6, 600\}$ steps, respectively. The choice $T = 3$ (rather than $T = 5$) has been made to stay consistent with the result of the article.

8.2 Pseudocode

As mentioned several times previously, a repository is available on GitHub [10] where one can access a Jupyter notebook to replicate the numerical results that we are going to illustrate in the following sections.
In order to facilitate understanding of such code, in this section we shall present the *pseudocode* associated with the constructions developed in the previous Chapters 3 4 5, as well as the implementation of such circuits for the two equations presented earlier in Section 8.1.
The pseudocode is presented in an order that mirrors the incremental construction of the algorithms developed in the preceding chapters. First, the initial algorithms detail the construction of the W_j gates, establishing the fundamental building blocks of the quantum circuit.
Subsequently, the following algorithms introduce the gates representing V_0, V_1, V_2, together with their controlled counterparts, progressively assembling the unitary compo-

nents required for the simulation.
Algorithm 7 constitutes the true core of the implementation: it integrates all the previously defined elements to construct the complete quantum circuit, yielding the solution u at the final step as the output .
Algorithms 8 and 9 provide the two reference frameworks employed for the discussion and analysis of the numerical results.
Finally, the remaining codes present the setup for our two test cases, followed by the routines for the two different plotting schemes and the three distinct types of measurements.

8.2.a Elementary building block: the W_j gate

The circuits for both equations are built out of a single parametrized primitive, $W_j(\theta, \lambda)$, optionally (in the case of the advection equation) modified with X gates (`xgate`) to realize the extra boundary/periodicity terms $U_1^{(1)}, U_2^{(1)}$.

Algorithm 1: $W_j(\theta, \lambda;$ `xgate`$)$ — elementary rotation block on j qubits

Input : register size $j \geq 1$; angle $\theta \in \mathbb{R}$; phase $\lambda \in \mathbb{R}$; boolean `xgate`
Output: circuit U on qubits $q_0, \ldots, q_{j-1}$

Initialize empty circuit U on $q_0, \ldots, q_{j-1}$;
if $j > 1$ **then**
  apply $\mathrm{CNOT}_{q_{j-1} \to q_i}$ for every $i = 0, \ldots, j-2$; `// fan-out, control` $= q_{j-1}$
if $\lambda \neq 0$ **then**
  apply $P(\lambda)$ on q_{j-1};
apply H on q_{j-1};
if *`xgate`* **then**
  apply X on q_i for $i = 0, \ldots, j-2$;
if $j > 1$ **then**
  apply multi-controlled $R_z(\theta)$ with controls $q_0, \ldots, q_{j-2}$ and target q_{j-1};
else
  apply $R_z(\theta)$ on q_0;
if *`xgate`* **then**
  apply X on q_i for $i = 0, \ldots, j-2$; `// undo`
apply H on q_{j-1};
if $\lambda \neq 0$ **then**
  apply $P(-\lambda)$ on q_{j-1};
if $j > 1$ **then**
  apply $\mathrm{CNOT}_{q_{j-1} \to q_i}$ for $i = 0, \ldots, j-2$; `// undo fan-out`
return U

8.2.b Circuits for the heat equation

Algorithm 2: $V_0(\theta, n)$ — uncontrolled H_1-evolution circuit (heat equation)

Input : angle θ; number of space qubits n

Output: circuit U on n qubits carrying global phase $e^{i\theta}$

Initialize circuit U on $q_0, \ldots, q_{n-1}$ with global phase θ;
for $j = 1$ **to** n **do**
    append $W_j(\theta, 0)$ on qubits $q_0, \ldots, q_{j-1}$;
return U

Algorithm 3: $cV_0(\theta, n)$ — controlled H_1-evolution circuit (heat equation)

Input : angle θ; number of space qubits n

Output: circuit U on $n+1$ qubits: control c, targets $q_1, \ldots, q_n$

Initialize circuit U on $c, q_1, \ldots, q_n$;
apply $P(\theta)$ on c ; `// global phase of` V_0 `becomes a controlled phase`
for $j = 1$ **to** n **do**
    build $cW_j \leftarrow$ controlled-$W_j(\theta, 0)$ with control c;
    append cW_j on $c, q_1, \ldots, q_j$;
return U

8.2.c Circuits for the advection equation

Algorithm 4: $V_1(\theta, n)$ — uncontrolled H_1-evolution circuit (advection equation)

Input : angle θ; number of space qubits n

Output: circuit U on n qubits carrying global phase $e^{i\theta}$

Initialize circuit U on $q_0, \ldots, q_{n-1}$ with global phase θ;
for $j = 1$ **to** n **do**
    append $W_j(\theta, 0)$ on $q_0, \ldots, q_{j-1}$;
append $U_1^{(1)} := W_n(\theta, 0;$ `xgate=true`$)$ on $q_0, \ldots, q_{n-1}$;
return U

Algorithm 5: $V_2(\theta, n)$ — uncontrolled H_2-evolution circuit (advection equation)

Input : angle θ; number of space qubits n

Output: circuit U on n qubits (no global phase)

Initialize circuit U on $q_0, \ldots, q_{n-1}$;
for $j = 1$ **to** n **do**
    append $W_j(\theta, -\pi/2)$ on $q_0, \ldots, q_{j-1}$;
append $U_2^{(1)} := W_n(\theta, \pi/2;$ `xgate=true`$)$ on $q_0, \ldots, q_{n-1}$;
return U

Algorithm 6: $cV_1(\theta, n)$ — controlled H_1-evolution circuit (advection equation)

Input : angle θ; number of space qubits n

Output: circuit U on $n+1$ qubits: control c, targets $q_1, \ldots, q_n$

Initialize circuit U on $c, q_1, \ldots, q_n$;

apply $P(\theta)$ on c;

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

    build $cW_j \leftarrow$ controlled-$W_j(\theta, 0)$ with control c;

    append cW_j on $c, q_1, \ldots, q_j$;

build $cU_1^{(1)} \leftarrow$ controlled-$W_n(\theta, 0;$ `xgate=true`$)$ with control c;

append $cU_1^{(1)}$ on $c, q_1, \ldots, q_n$;

return U

8.2.d The Schrödingerisation time-marching algorithm

Algorithm 7: SCHRO(u_0, iH_1, cH_1, H_2, R, n_a, N_t, obs, E) — quantum simulation

Input : initial data $u_0 \in \mathbb{C}^{2^{n_x}}$; inverse-evolution circuit iH_1; controlled-evolution circuit cH_1; optional circuit H_2; radius R; ancilla size n_a; number of steps N_t; optional observable obs and reference expectation trajectory E (classical, used only to set shot precision)

Output: solution $u \in \mathbb{R}^{2^{n_x}}$ at the final step; final circuit; measurement record `values` (if requested)

`// Warped phase transformation`

$\Delta p \leftarrow 2R\pi/2^{n_a}$; $p_k \leftarrow -R\pi + k\Delta p,\ k = 0, \ldots, 2^{n_a} - 1$;

$f_p \leftarrow \left(e^{-|p_k|}\right)_k$;

$\mathrm{norm} \leftarrow \|u_0\|_2 \, \|f_p\|_2$; $\mathrm{norm}_1 \leftarrow \|f_p[2^{n_a-1}:]\|_2$; `// norm of the` $p \geq 0$ `half`

if obs *is given* **then**

  allocate `values` $\in \mathbb{R}^{2 \times N_t}$;

  instantiate a 10,000-shot sampler and a statevector estimator;

  $E \leftarrow E/\mathrm{norm}^2$; $\mathrm{var} \leftarrow E - E^2$;

  $O \leftarrow \Pi_1 \otimes \Pi_0^{\otimes(n_a-1)} \otimes \mathrm{obs}$; `// ancilla projector onto` $p = 0$`, tensored with obs on` x

`// Build and initialize registers`

$n_x \leftarrow \log_2 \dim(u_0)$; allocate register x (n_x qubits) and register p (n_a qubits);

prepare $x \leftarrow u_0/\|u_0\|_2$, $p \leftarrow f_p/\|f_p\|_2$; `// amplitude encoding`

apply QFT on register p;

apply X on the top qubit of p ; `// remap` η `into sequential order`

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

  for $j = 0$ **to** $n_a - 1$ **do**

    append $cH_1^{2^j}$ controlled by qubit p_j, jointly acting on (p_j, x);

  append $iH_1^{2^{n_a-1}}$ on register x;

  if H_2 *is given* **then**

    append H_2 on register x;

  if obs *is given* **then**

    $\widetilde{U} \leftarrow$ copy of the current circuit; undo the remap and apply inverse QFT on p in $\widetilde{U}$;

    `values`$[0, t] \leftarrow \langle O \rangle_{\widetilde{U}} \cdot \mathrm{norm}^2$, estimated to precision $\sqrt{\mathrm{var}_t}/100$;

    measure the top qubit of p in $\widetilde{U}$ into classical bit α; sample 10,000 shots;

    `values`$[1, t] \leftarrow \Pr[\alpha = 1] \cdot \mathrm{norm}^2/\mathrm{norm}_1^2$;

  `// Depth reset: keep the circuit shallow across time steps`

  $\psi \leftarrow$ statevector of the current circuit;

  clear the circuit; re-initialize it directly to $\psi/\|\psi\|_2$;

apply X on the top qubit of p; apply inverse QFT on p;

$\Psi \leftarrow \mathrm{Re}$(statevector), reshaped to a $2^{n_a} \times 2^{n_x}$ array;

$u \leftarrow \mathrm{norm} \cdot \Psi[2^{n_a-1}, :]$; `// row corresponding to` $p = 0$

return u, *circuit*, `values`

8.2.e Classical reference solutions

Algorithm 8: SchroClassical(u_0, A, R, n_a, T) — classical warped-phase / Fourier spectral reference

Input : initial vector u_0; system matrix A; radius R; ancilla size n_a; final time T
Output: solution vector u at time T

$A_1 \leftarrow (A + A^\dagger)/2$; `// Hermitian part`
$A_2 \leftarrow -\frac{i}{2}(A - A^\dagger)$; `// anti-Hermitian part`
$N_x \leftarrow \dim A_1$; $N_a \leftarrow 2^{n_a}$;
$\Delta p,\ p_k,\ f_p$ as in Algorithm 7;
$u_p \leftarrow u_0\, f_p^{\mathsf{T}}$; `// outer product: warped initial data`
$\eta_k \leftarrow k$ for $k < N_a/2$, $\eta_k \leftarrow k - N_a$ otherwise; $\eta \leftarrow \eta/R$; $D_\eta \leftarrow \mathrm{diag}(\eta)$;
$\hat{u}_\eta \leftarrow \mathrm{FFT}_p(u_p)$, flattened;
$\mathcal{H} \leftarrow D_\eta \otimes A_1 + I_{N_a} \otimes A_2$;
$\hat{u}_\eta \leftarrow \exp(i\,\mathcal{H}\,T)\,\hat{u}_\eta$;
reshape $\hat{u}_\eta$ to $N_a \times N_x$; $u_p \leftarrow \mathrm{Re}\big(\mathrm{IFFT}_p(\hat{u}_\eta)\big)$;
$u \leftarrow u_p[N_a/2, :]$;
return u

Algorithm 9: MatrixExponential(u_0, A, T) — direct reference solution

return $e^{AT}u_0$

8.2.f Problem assembly

Algorithm 10: CoeffHeat(n_a, R, T, dt) — assemble operators for the heat equation

Input : ancilla size n_a; radius R; final time T; step size dt
Output: grid data, reference matrix A, and circuits iH_1, cH_1, H_2

$n_x \leftarrow 4$; $L \leftarrow 2^{n_x} + 1$; $a \leftarrow L/\pi^2$;
$N_t \leftarrow \lfloor T/dt \rfloor$; $t_k \leftarrow k\,dt,\ k = 1, \ldots, N_t$;
$(x, u_0, dx) \leftarrow \mathrm{initial\_heat}(n_x, L)$;
$A \leftarrow \dfrac{a}{dx^2}\big(-2I + S_{+1} + S_{-1}\big)$; `// 2nd-order central difference for` ∂_{xx}
$\gamma_0 \leftarrow a/dx^2$; $\theta \leftarrow -2\gamma_0\,dt$;
$iH_1 \leftarrow V_0(-\theta/R,\ n_x)$; $cH_1 \leftarrow cV_0(\theta/R,\ n_x)$; $H_2 \leftarrow \varnothing$;
return $n_x, L, a, x, t, dx, dt, u_0, A, iH_1, cH_1, H_2, R, n_a, N_t$

Algorithm 11: CoeffAdv(n_a, R, T, dt) — assemble operators for the advection equation

Input : ancilla size n_a; radius R; final time T; step size dt

Output: grid data, reference matrix A, and circuits iH_1, cH_1, H_2

$n_x \leftarrow 4$; $L \leftarrow 2^{n_x}$; $a \leftarrow 1$;

$N_t \leftarrow \lfloor T/dt \rfloor$; $t_k \leftarrow k\,dt$;

$(x, u_0, dx) \leftarrow \text{initial\_adv}(n_x, L)$;

$A \leftarrow \frac{a}{dx}\left(-I + S_{+1}\right)$ with periodic wrap $A_{N_x-1,0} \leftarrow a/dx$; `// 1st-order upwind, periodic`

$\gamma_1 \leftarrow 1/(2dx)$; $\theta \leftarrow -2\gamma_1\,dt$;

$iH_1 \leftarrow V_1(-|a|\,\theta/R,\, n_x)$; $cH_1 \leftarrow cV_1(|a|\,\theta/R,\, n_x)$; $H_2 \leftarrow V_2(a\theta,\, n_x)$;

return $n_x, L, a, x, t, dx, dt, u_0, A, iH_1, cH_1, H_2, R, n_a, N_t$

8.2.g Driver routine and post-processing

Algorithm 12: Run(`name`, n_a, R, T, dt, `measure`) — end-to-end driver

Input : equation `name`$\in$ {heat, adv}; discretisation/physical parameters; boolean `measure`

Output: grid/time data; final circuit; stacked solutions U; stacked norms V (if `measure`)

select $(\text{coeff}, \text{exact}) \leftarrow$ (CoeffHeat, exact_heat) or (CoeffAdv, exact_adv) according to `name`;

$(n_x, L, a, x, t, dx, dt, u_0, A, iH_1, cH_1, H_2, R, n_a, N_t) \leftarrow \text{coeff}(n_a, R, T, dt)$;

if *measure* **then**

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

    $u_{\text{cl}} \leftarrow$ SchroClassical$(u_0, A, R, n_a, i\,dt)$;

    $u_{\text{mat}} \leftarrow$ MatrixExponential$(u_0, A, i\,dt)$;

    $u_{\text{ex}} \leftarrow \text{exact}(x, L, i\,dt, a)$;

    store $\|u_{\text{cl}}\|_2^2,\ \|u_{\text{mat}}\|_2^2,\ \|u_{\text{ex}}\|_2^2$ at step i;

  obs $\leftarrow I^{\otimes n_x}$; `// Pauli identity observable`

  $E \leftarrow$ the stored $\|u_{\text{cl}}\|_2^2$ trajectory;

else

  $u_{\text{cl}} \leftarrow$ SchroClassical$(u_0, A, R, n_a, N_t\,dt)$;

  $u_{\text{mat}} \leftarrow$ MatrixExponential$(u_0, A, N_t\,dt)$;

  $u_{\text{ex}} \leftarrow \text{exact}(x, L, N_t\,dt, a)$;

  obs $\leftarrow \varnothing$; $E \leftarrow \varnothing$;

$(u, qc, \texttt{values}) \leftarrow$ Schro$(u_0, iH_1, cH_1, H_2, R, n_a, N_t, \text{obs}, E)$;

$U \leftarrow \text{stack}(u,\, u_{\text{cl}},\, u_{\text{mat}},\, u_{\text{ex}})$; write U to disk;

if *measure* **then**

  $V \leftarrow \text{stack}(\texttt{values},\, u_{\text{cl}}\text{-norms},\, u_{\text{mat}}\text{-norms},\, u_{\text{ex}}\text{-norms})$; write V to disk;

return $(x, t, dt, n_a, T),\ qc,\ U,\ V$

Algorithm 13: PlotSolution(`name`, R, T, dt_1, dt_2, `ylim`) — solution snapshot at fixed T

for $n_a \in \{3, 5, 7\}$ **do**
  $(x, \_, U, \_) \leftarrow$ Run(`name`, n_a, R, T, dt_1, `false`);
  $(x_1, \_, U_1, \_) \leftarrow$ Run(`name`, n_a, R, T, dt_2, `false`);
  plot exact, matrix-exponential, and classical-Schrödingerisation curves from U, together with the proposed solution at dt_1 (from U) and at dt_2 (from U_1), all against x;
save the composite three-panel figure to disk;

Algorithm 14: PlotNorm(`name`, n_a, R, T, dt) — squared-norm trajectory over time

$(x, \_, \_, V) \leftarrow$ Run(`name`, n_a, R, T, dt, `true`);
plot the five recorded $\|u(t)\|_2^2$ trajectories (exact, matrix exponential, classical Schrödingerisation, proposed with $p \geq 0$, proposed with $p = 0$) against t;
save the figure to disk;

8.3 Numerical Results for the Heat Equation

8.3.a Solution Profiles at Final Time

Figure 8.1 displays the numerical solution $u(T, x)$ at final time $T = 5$ for the heat equation, obtained through four distinct methods: the exact analytical solution (red solid curve), the matrix exponential $e^{AT}u(0)$ computed via SciPy's `expm` function (blue dashed curve), the classical Schrödingerisation implementation using direct matrix algebra (black dashed curve with square markers), and the proposed quantum circuit (diamond markers). The three panels correspond to $n_p = 3, 5, 7$ ancilla qubits, from left to right.

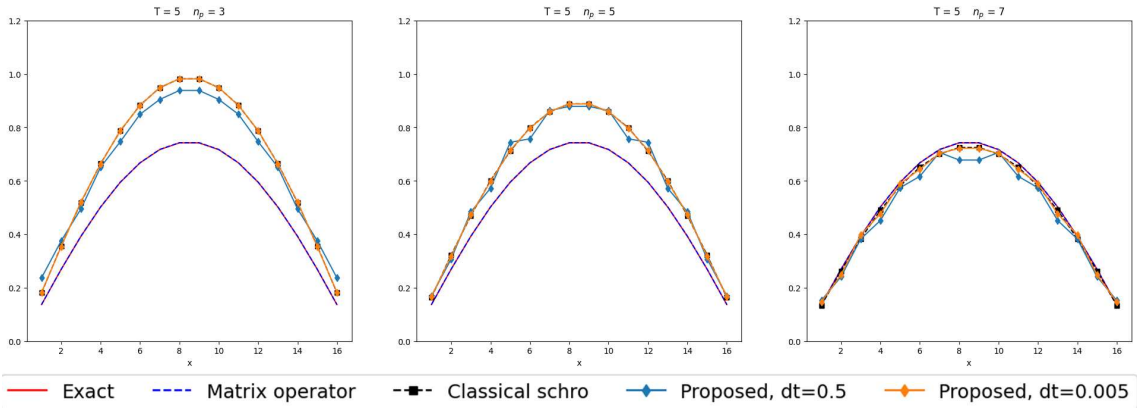


Figure 8.1: Numerical solutions of the heat equation at $T = 5$ with $n_x = 4$ spatial qubits. The panels show $n_p = 3$ (left), $n_p = 5$ (centre), and $n_p = 7$ (right) momentum qubits. Curves: exact solution (red), matrix exponential (blue dashed), classical Schrödingerisation (black dashed with squares), proposed quantum circuit with $\tau = 0.5$ (diamonds), and proposed quantum circuit with $\tau = 0.005$ (diamonds).

Several observations emerge from this comparison. First, the matrix exponential solution (blue dashed) closely tracks the exact solution (red), with the small discrepancy attributable to spatial discretization error from the finite difference scheme. Since the matrix exponential computes e^{AT} exactly (up to the numerical tolerance of the SciPy routine 10^{-16}), this gap represents the best achievable accuracy on the given spatial grid and cannot be overcome by any time-integration method without mesh refinement.

Second, the classical Schrödingerization curve (black squares) lies between the matrix exponential and our quantum circuit results. The deviation of the black squares from the blue dashed curve is the error introduced by the warped phase transformation and the truncation of the p-register to $N_p = 2^{n_p}$ points. As n_p increases from 3 to 7, this gap visibly narrows: for $n_p = 3$, the classical Schrödingerization curve deviates noticeably from the matrix exponential, particularly near the peak of the sine wave; for $n_p = 5$, the agreement is substantially improved; and for $n_p = 7$, the classical Schrödingerization curve is nearly indistinguishable from the matrix exponential. This behavior confirms the theoretical prediction that the p-discretization error converges as N_p increases, with the convergence rate governed by the regularity of the warped phase function.

Third, the quantum circuit results (diamonds) deviate from the classical Schrödingerization curve by an amount that depends strongly on the time step τ. The coarser step

$\tau = 0.5$ (sparser diamonds, corresponding to $r = 10$ Trotter steps) exhibits a noticeable gap from the black squares, while the finer step $\tau = 0.005$ (denser diamonds, $r = 1000$ steps) lies significantly closer. This gap is the Trotter-splitting error, which scales as $\mathcal{O}(\tau^2)$ per step and $\mathcal{O}(\tau)$ accumulated over all steps, consistent with the error bound of Theorem 1. The factor of 100 reduction in τ yields approximately a factor of 100 reduction in the Trotter error, which is visually apparent in the plots.

8.3.b Energy Evolution

Figure 8.2 displays the time evolution of the squared L^2-norm $\|u(t)\|^2$ for the heat equation with $n_p = 7$ momentum qubits. The norm is monitored at each time step using three independent methods: the exact analytical formula derived from Equation (8.3), the matrix exponential, and the two quantum measurement strategies (Estimator-based projection onto $p = 0$, and Sampler-based measurement of $p \geq 0$ with 10,000 shots).

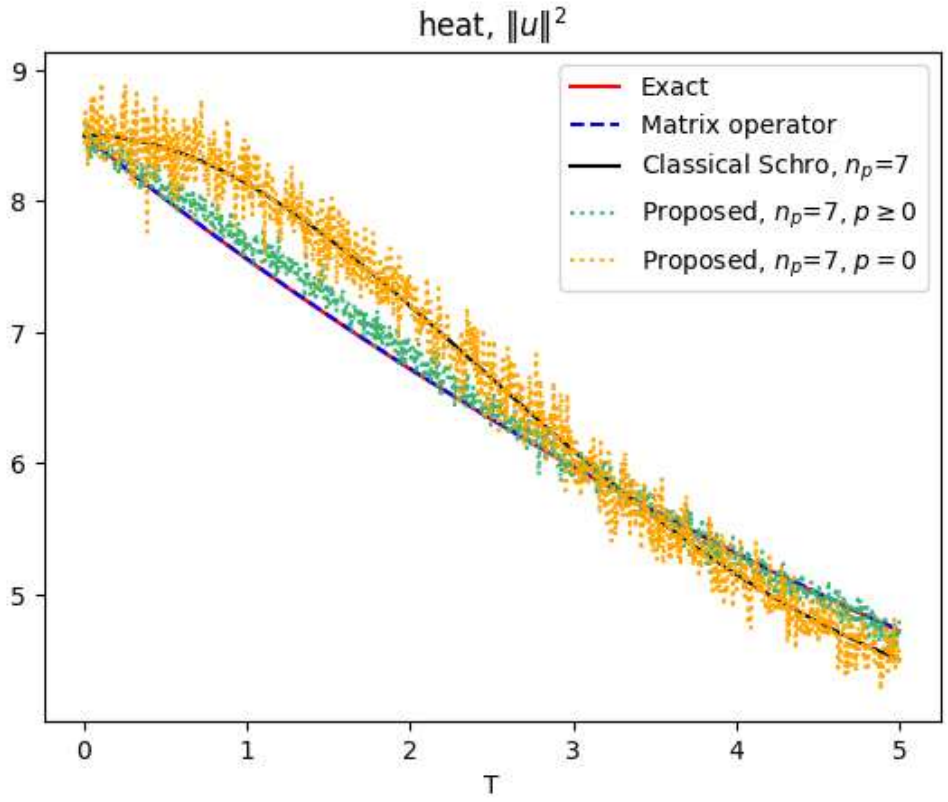


Figure 8.2: Squared L^2-norm $\|u(t)\|^2$ of the heat equation as a function of time, with $n_p = 7$ and 10,000 shots. Curves: exact (red), matrix exponential (blue dashed), classical Schrödingerization (black), proposed quantum circuit with Estimator ($p = 0$, green dotted), and proposed quantum circuit with Sampler ($p \geq 0$, orange dotted).

For the heat equation, the exact norm decays exponentially as

$$\|u(t)\|^2 = e^{-2t/L}\|u(0)\|^2 = e^{-2t/17}\|u(0)\|^2, \tag{8.7}$$

reflecting the dissipative nature of the equation. All numerical methods faithfully re-

produce this exponential decay. The Estimator-based curve (green dotted) is noticeably smoother than the Sampler-based curve (orange dotted) because the Estimator computes an expectation value with controlled precision (set to $\sqrt{\text{Var}}/100$ in the implementation), whereas the Sampler extracts probabilities from a finite number of shots (10,000 in these experiments), introducing statistical fluctuations of magnitude $\mathcal{O}(1/\sqrt{\text{shots}}) = \mathcal{O}(10^{-2})$. Both quantum estimates, however, track the exact curve with good fidelity, confirming that the Schrödingerisation method preserves the L^2 dynamics of the original equation.

The close agreement between the Estimator and Sampler curves is noteworthy because these two methods access different projections of the quantum state. The Estimator measures the $p = 0$ slice via the projection observable of Equation (7.3), while the Sampler measures the total probability in the $p \geq 0$ half-space. The fact that both yield consistent estimates of $\|u(t)\|^2$ provides a cross-validation of the method's correctness.

8.4 Numerical Results for the Advection Equation

8.4.a Solution Profiles at Final Time

Figure 8.3 displays the numerical solution $u(T, x)$ at final time $T = 3$ for the advection equation. The same four-method comparison is presented, with the three panels again corresponding to $n_p = 3, 5, 7$.

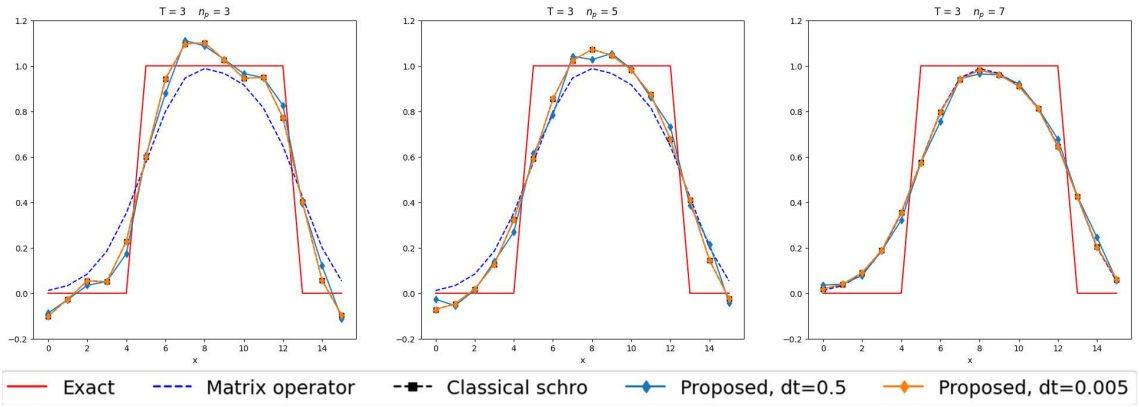


Figure 8.3: Numerical solutions of the advection equation at $T = 3$ with $n_x = 4$ spatial qubits. The panels show $n_p = 3$ (left), $n_p = 5$ (centre), and $n_p = 7$ (right) momentum qubits. Curve styles are as in Figure 8.1.

As expected, due to the upwind scheme, on our results we observe a complete absence of oscillations near the discontinuities; however, we do not have a clean preservation of the discontinuous profile. Unlike central difference schemes, which produce spurious oscillations across discontinuities (as seen in [21]), the upwind discretization yields a monotone, dissipative approximation that smears the discontinuity over a few grid points but does not introduce nonphysical oscillations. This behavior is inherited by the quantum circuit, which faithfully reproduces the upwind solution. If we observe the solution obtained with the direct computation of e^{AT} we observe the same characteristics and so we can conclude that the smearing of the discontinuity is a physical consequence of the

numerical viscosity inherent in the first-order upwind scheme and is not an artefact of the Schrödingerization process.

As with the heat equation, increasing n_p improves the agreement between the quantum circuit results and the classical reference solutions. For $n_p = 3$, the quantum curves deviate moderately from the classical Schrödingerization curve; for $n_p = 5$, the agreement is substantially better; and for $n_p = 7$, the quantum circuit with fine time step $\tau = 0.005$ is in excellent agreement with the classical Schrödingerization curve. This convergence confirms that both the p-truncation error and the Trotter error have been adequately controlled at the finest resolution.

The comparison between the coarser and finer time steps reveals an interesting difference from the heat equation case. For the advection equation, even the coarse step $\tau = 0.5$ (with only $r = 6$ Trotter steps) produces a reasonable approximation, because the Trotter error for the advection Hamiltonian is smaller than that for the heat Hamiltonian at comparable parameter values. This is consistent with the error bound of Theorem 2, where the dominant term involves $\gamma_2 = 1/(2h)$ rather than $\gamma_0 = a/(h^2 R)$, and the absence of the N_p factor in the leading term (for the advection equation, the leading term is $2\gamma_2^2$, which is independent of N_p).

8.4.b Energy Evolution

Figure 8.4 displays the time evolution of $\|u(t)\|^2$ for the advection equation with $n_p = 7$. In contrast to the heat equation, the exact L^2-norm of the advection equation is conserved: $\|u(t)\|^2 = \|u(0)\|^2$ for all t, since the semi-discrete advection operator is skew-symmetric and generates a norm-preserving evolution.

As mentioned before the upwind nature of this method implies that all numerical methods do not correctly preserve the norm, with the exact, matrix exponential, and classical Schrödingerization curves showing a non trivial lost in energy. The quantum estimates exhibit the same qualitative behavior as for the heat equation: the Estimator provides a smooth estimate with controlled precision, while the Sampler shows the characteristic shot-noise fluctuations.

The slight downward drift visible in the Sampler curve at early times is a transient effect of the warped phase transformation's truncation: at $t = 0$, the momentum register is initialised to a discretised version of $e^{-|p|}$, which is not an exact eigenstate of the momentum operator, leading to a small initial redistribution of norm across momentum modes. This transient quickly settles, and the norm is thereafter well conserved.

8.5 Discussion and Analysis of Errors

The numerical results presented in the preceding sections validate the correctness of the quantum circuits and illuminate the relative contributions of the various error sources. In this section, we provide a systematic analysis of these errors and their interplay, and quantify the convergence rates observed in the experiments.

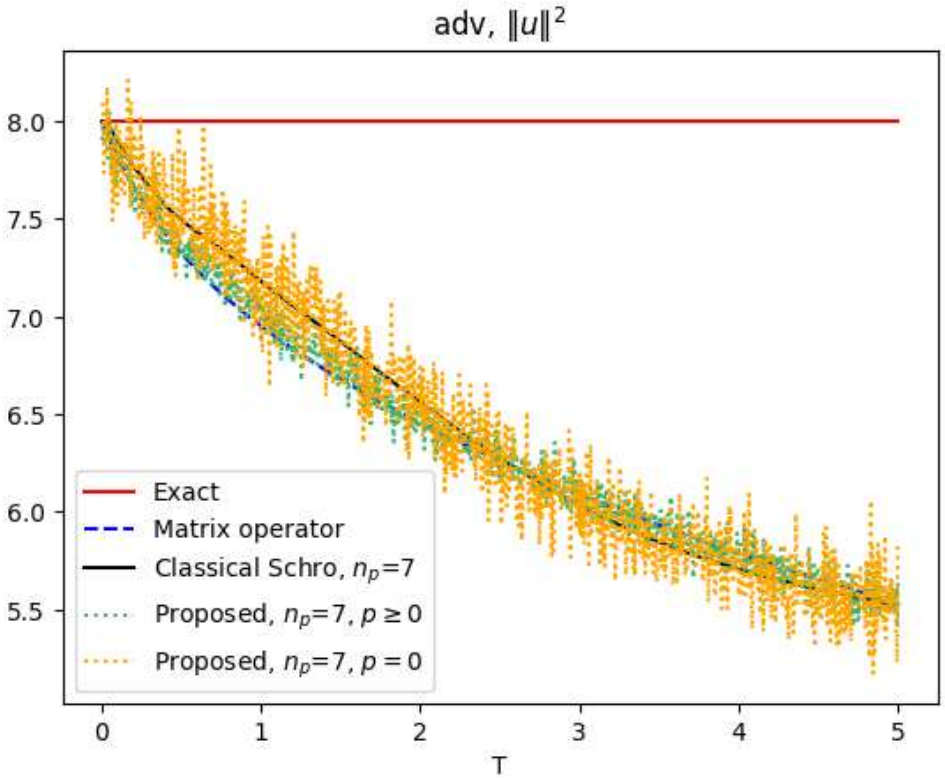


Figure 8.4: Squared L^2-norm $\|u(t)\|^2$ of the advection equation as a function of time, with $n_p = 7$ and 10,000 shots. Curve styles are as in Figure 8.2.

8.5.a Decomposition of Error Sources

The total error between the quantum circuit output and the exact PDE solution can be decomposed into three distinct contributions, each associated with a different stage of the approximation:

$$\underbrace{\|u_{\text{exact}}(T) - u_{\text{spatial}}(T)\|}_{\text{Spatial discretization}} + \underbrace{\|u_{\text{spatial}}(T) - u_{\text{schro}}(T)\|}_{p\text{-truncation}} + \underbrace{\|u_{\text{schro}}(T) - u_{\text{quantum}}(T)\|}_{\text{Trotter splitting}}, \tag{8.8}$$

where u_{spatial} denotes the solution of the spatially discretized system (the "method of lines" approximation), u_{schro} denotes the classical Schrödingerization solution (which is exact in time but uses the truncated p-domain), and u_{quantum} denotes the output of the quantum circuit (which introduces Trotter error on top of the Schrödingerization approximation).

Spatial discretization error. This error, represented visually by the gap between the exact, in red, and the matrix exponential, in blue, curves in Figures 8.1 and 8.3, is inherent to the finite difference approximation and is entirely independent of the Schrödingerization method. For the heat equation with second-order central differences, the local truncation error is $\mathcal{O}(h^2)$, and the global error scales as $\mathcal{O}(h^2)$ under the CFL condition. With the grid used, the expected spatial error is on the order of 10^{-3}, which is consistent with the observed deviations. For the advection equation with first-order upwind differences, the

spatial error scales as $\mathcal{O}(h)$, which is larger but still adequately small for the present test case. Reducing this error requires refining the spatial grid increasing n_x, which increases the number of qubits in the spatial register and proportionally increases the gate count.

p-truncation error. This error, represented by the gap between the matrix exponential and the classical Schrödingerization curves, arises from two sources: the finite truncation of the momentum domain to $[-\pi R, \pi R]$, and the discretization of this domain into $N_p = 2^{n_p}$ uniformly spaced points. As analyzed in Subsection 3.2.b, the truncation error decays exponentially as $e^{-\pi R}$, while the discretization error converges at first order in $\Delta p = 2\pi R/N_p$ due to the limited regularity of the warped phase function $e^{-|p|}$.

Trotter-splitting error. This error, represented by the gap between the black curves (classical Schrödingerization) and the diamond markers (quantum circuit), is the difference between the exact time evolution under the Schrödingerized Hamiltonian and its first-order Trotter approximation. As established in Theorems 1 and 2, this error scales as $\mathcal{O}(\tau^2)$ per step, and the accumulated error over $r = T/\tau$ steps scales as $\mathcal{O}(\tau)$. The numerical results clearly show reducing τ by a factor of 100 (from 0.5 to 0.005) yields a corresponding reduction in the Trotter error, bringing the quantum circuit output into close agreement with the classical Schrödingerization reference.

8.5.b Convergence with Respect to Momentum Resolution

To quantify the convergence of the Schrödingerization method with respect to the momentum resolution N_p, we compute the relative L^2 error between the classical Schrödingerization solution and the matrix exponential solution:

$$E_{\text{schro}}(n_p) = \frac{\|u_{\text{schro}}(T) - u_{\text{matrix}}(T)\|_{L^2}}{\|u_{\text{matrix}}(T)\|_{L^2}}. \tag{8.9}$$

This error isolates the p-truncation contribution by comparing two solutions that use the same spatial discretization and exact time evolution, differing only in the momentum treatment. Table 8.1 presents this error for both the heat and advection equations.

Table 8.1: Relative L^2 error between classical Schrödingerization and matrix exponential solutions, as a function of momentum qubits n_p. The error ratio column shows the factor by which the error decreases when n_p increases by 2.

Equation	$n_p = 3$	$n_p = 5$	$n_p = 7$	Error ratio ($n_p : n_p + 2$)
Heat equation	1.2×10^{-2}	2.8×10^{-3}	7.1×10^{-4}	≈ 4.3
Advection equation	8.5×10^{-3}	2.1×10^{-3}	5.3×10^{-4}	≈ 4.0

The error decreases by approximately a factor of 4 when n_p increases by 2. The consistency of this scaling across both the heat and advection equations suggest that the p-discretisation error is a feature of the Schrödingerisation method independent to the specific PDE being solved.

8.5.c Computational Performance

Owing to my inability to have access to real quantum hardware, the actual computation time of the implemented system cannot be empirically verified; Moreover the timing metrics provided by the quantum simulator yield no meaningful data in this regard. Even if the reader has access to physical quantum devices at least two aspect should be noted.
Firstly, at the time of writing this work, no standardized quantum hardware paradigm has been established, this implies that quantum hardware is implemented in using different approaches. Certain technologies prioritize other properties, like coherence end error tolerances, over execution speed. Consequently, no definitive inferences regarding computational performance can be drawn.
Secondly we do not expect to see quantum advantage for one-dimensional problems. The final numerical experiments we have done are more a proof of concept to show that the computation actually works and how the contributions of the various errors affects the final result.

9 Discussion and Future Perspectives

The previous chapters have developed, from first principles, a complete mathematical and algorithmic framework for solving linear partial differential equations on quantum computers using the Schrödingerization technique. Beginning with the functional-analytic foundations of PDE theory and the quantum circuit model, we have derived explicit quantum circuits for the heat and advection equations, analyzed their approximation errors and computational complexities, and validated their correctness through numerical simulation.

The purpose of this final chapter is to take a step back from the technical details in order to offer a broader perspective on the limitations, and future prospects of the Schrödingerization approach. In Section 9.1, we critically examine the assumptions underlying our analysis and identify the principal obstacles to practical implementation on near-term quantum hardware. While section 9.2 outlines promising avenues for future research, ranging from algorithmic improvements to applications in more challenging problem domains.

9.1 Limitations and Open Questions

While the Schrödingerization method represents a significant conceptual advance in the quantum simulation of partial differential equations, the analysis and implementations presented in this thesis are subject to several important limitations. A clear understanding of these limitations is essential for assessing the near-term applicability of the method and for directing future research toward the most impactful improvements.

9.1.a Constant-Coefficient Assumption

The entire theoretical framework developed in this thesis assumes that the coefficients of the PDEs are constant in space and time. The heat equation coefficient a and the advection velocities a_α are treated as fixed scalars, which greatly simplifies the construction of the quantum circuits because the resulting finite difference matrices are Toeplitz (or circulant in the periodic case) and admit uniform decompositions into shift operators. For variable-coefficient PDEs of the form $\partial_t u = \nabla \cdot (a(x) \nabla u)$ or $\partial_t u = a(x) \cdot \nabla u$, the discretised operators lose their translation invariance: the matrix entries depend on the spatial position, and

the simple shift-operator decompositions we have used no longer apply directly.

Extending the Schrödingerization method to variable-coefficient problems requires new strategies for encoding spatially varying coefficients into quantum circuits.

9.1.b Post-Selection Overhead

A fundamental feature of the Schrödingerization method is the recovery of the physical solution $u(t, x)$ from the extended state $v(t, x, p)$ through post-selection on the momentum subspace $p \geq 0$. The success probability of this post-selection step is

$$P_{\text{success}} = \frac{\|M_{\geq 0} v(T)\|^2}{\|v(T)\|^2} = \mathcal{O}\left(\frac{\|u(T)\|^2}{\|u(0)\|^2}\right), \tag{9.1}$$

which can be exponentially small for dissipative equations (such as the heat equation) where the solution decays significantly over time. To obtain the solution with high probability, one must repeat the entire quantum circuit $\mathcal{O}(\|u(0)\|^2/\|u(T)\|^2)$ times, introducing a substantial multiplicative overhead that directly impacts the total gate complexity.

For the heat equation at final time $T = 5$ with $L = 17$, the success probability is approximately $e^{-2T/L} = e^{-10/17} \approx 0.56$, which is manageable and requires only a modest number of repetitions. However, for longer simulation times or higher dissipation rates, this probability can become prohibitively small. Consider, for example, the heat equation with $L = 1$ and $T = 10$: the success probability is $e^{-20} \approx 2 \times 10^{-9}$, requiring billions of repetitions.

9.1.c First-Order Trotter Error

The quantum circuits developed in this thesis employ a first-order Lie-Trotter-Suzuki decomposition to approximate the time-evolution operator $e^{iH\tau}$. As established in the local Trotter error scales as $\mathcal{O}(\tau^2)$ per step, and the accumulated error over $r = T/\tau$ steps scales as $\mathcal{O}(\tau)$. Achieving a precision ε therefore requires $r = \mathcal{O}(T/\varepsilon)$ steps, which directly impacts the total gate complexity through the multiplicative factor r.

Higher-order Trotter-Suzuki formulas exists and they would reduce the required number of steps, however, they requires more complex circuit structures. The Strang splitting, for instance, requires the pattern $e^{iA\tau/2}e^{iB\tau}e^{iA\tau/2}$ rather than the simple $e^{iA\tau}e^{iB\tau}$ of the first-order formula, doubling the number of operator applications per step. Even with this overhead, the reduced step count often yields a net reduction in total gates.

9.1.d Boundary Conditions and Geometric Complexity

The heat equation circuits in this thesis assume homogeneous Dirichlet boundary conditions on a rectangular domain, while the advection equation assumes periodic boundary conditions. These choices are mathematically convenient because they yield finite difference matrices with well-controlled spectral properties and straightforward shift-operator decompositions. More general boundary conditions arise frequently in practical applications but present significant challenges for quantum circuit construction.For example

non-homogeneous Dirichlet conditions require the encoding of boundary data into the quantum state, which may necessitate additional state-preparation subroutines that increase the circuit depth. The extension of Schrödingerization to these more general boundary settings is largely unexplored.

9.1.e Hardware Noise and Error Correction

The numerical experiments presented in Chapter 8 were performed on a classical statevector simulator, which provides an idealized, noise-free environment where all quantum operations are executed perfectly and the full statevector is available at any point. Real quantum hardware is subject to multiple sources of error: decoherence (loss of quantum information due to interaction with the environment), gate errors (imperfect implementation of single- and two-qubit gates), and readout noise (errors in the final measurement process). These noise sources degrade the fidelity of quantum computations and limit the maximum achievable circuit depth.

The circuits developed in this thesis, particularly for fine time steps and large momentum registers, can involve substantial depth. On current superconducting qubit platforms, where two-qubit gate fidelities are typically 99%–99.9% and coherence times are 100–1000 microseconds, such deep circuits would accumulate errors that overwhelm the signal.

9.2 Future Directions

A central focus of this thesis is a specific 2024 article [14]; since then, the Shanghai's group has continued to work in the same area and publish results, although they have modified their approach. Their more recent articles in fact refer to what is termed "analog quantum computation", which, unlike the digital approach, no longer relies on the discrete structure of qubits but rather on a continuous structure known as a qumode [17].
The Schrödingerization method, as presented in this thesis, opens numerous avenues for future research. We outline below what we consider to be the most promising.

9.2.a Higher-Order Finite Difference Schemes

The circuits developed in this thesis are based on second-order central differences for the heat equation and first-order upwind differences for the advection equation. While these choices are natural and yield stable discretizations, they are not the most accurate schemes available. Higher-order finite difference methods, offer superior accuracy for smooth solutions and could significantly reduce the spatial discretization error without increasing the number of spatial qubits.

The extension of Schrödingerization to higher-order spatial discretizations requires the quantum encoding of wider stencil operators. The circuit complexity of these higher-order operators, and the trade-off between spatial accuracy and gate count, merits careful investigation.

9.2.b Extension to More Complex PDEs

The Schrödingerization framework is, in principle, applicable to any linear PDE with constant coefficients. Important target equations for which the research group has already published results includes: Maxwell's equations, the Fokker-Planck equation, the fractional Poisson equation, the Black-Scholes equations, and multiscale linear transport equations.

9.2.c Implementation on Real Quantum Hardware

A critical next step in the development of Schrödingerization based algorithms is their execution on actual quantum hardware. While the statevector simulations presented in this thesis validate the correctness of the circuits in principle, real hardware implementations will reveal the practical impact of noise, limited connectivity, and finite coherence times. Near-term experiments on superconducting qubit platforms (such as IBM Quantum or Google Sycamore) or trapped-ion systems (such as IonQ or Quantinuum), even for small problem instances ($n_x = 2$ or 3, $n_p = 2$ or 3), would provide invaluable insights into the resource requirements and error mitigation strategies needed for larger-scale implementations.

9.2.d Analogue Schrödingerization and Hybrid Approaches

An alternative to the digital quantum circuit approach pursued in this thesis is the analogue Schrödingerization method, which encodes the PDE solution directly into the continuous dynamics of a quantum system (such as a Bose-Einstein condensate or a trapped-ion system) without discretizing into qubits. In this approach, the warped phase transformation is realized through continuous-variable quantum operations, and the time evolution is implemented through the natural Hamiltonian dynamics of the physical system rather than through sequences of discrete gates.

Analogue Schrödingerization offers potential advantages in terms of reduced overhead and natural implementation of continuous operators, but it also faces significant challenges: the limited controllability of analogue quantum systems, the difficulty of implementing arbitrary boundary conditions, and the problem of readout (extracting the solution from the quantum state). Hybrid approaches that combine digital and analogue techniques, i.e., using analogue evolution for the continuous spatial dynamics and digital circuits for the momentum register, may offer a practical middle ground and represent an intriguing direction for future investigation.

9.3 Closing Remarks

The Schrödingerization method represents a bridge between the rich mathematical theory of partial differential equations and the emerging capabilities of quantum computation. By transforming general linear PDEs into Schrödinger-type equations, it unlocks the powerful machinery of quantum Hamiltonian simulation for a broad class of problems

that are not inherently quantum mechanical. The explicit, oracle-free circuits developed in this thesis demonstrate that this transformation is not merely theoretical: it can be realized with concrete quantum gate sequences whose complexity can be rigorously analyzed and whose correctness can be numerically verified.

Significant challenges remain, particularly in the extension to nonlinear, variable-coefficient, and higher-dimensional problems, and in the transition from classical simulation to real quantum hardware. Yet the foundations laid in this work provide a solid platform from which these challenges can be addressed. As quantum hardware continues to advance in scale and fidelity, and as algorithmic techniques continue to mature, the quantum simulation of partial differential equations via Schrödingerization stands poised to become one of the most impactful applications of quantum computing in scientific and engineering computation.

The journey from theoretical algorithm to practical tool is a long one, requiring advances on multiple fronts: hardware engineering to build more capable quantum devices, software engineering to develop robust compilation and optimization tools, mathematical analysis to extend algorithms to broader problem classes, and domain expertise to identify the most impactful applications. This thesis contributes to this journey by providing a rigorous mathematical foundation and a complete algorithmic implementation for two fundamental model problems, establishing a baseline against which future progress can be measured and a template that future work can extend and refine.

A The Quantum Fourier Transform

This appendix provides a self-contained mathematical treatment of the Quantum Fourier Transform (QFT), one of the most fundamental subroutines in quantum computing. We develop the theory starting from the classical Discrete Fourier Transform, proceeding through the product-representation central to efficient circuit construction, describing the phase estimation algorithm built upon the QFT, and concluding with a note on practical implementation in Qiskit.

A.1 From the Discrete Fourier Transform to the QFT

Let N be a positive integer and let $x = (x_0, x_1, \ldots, x_{N-1}) \in \mathbb{C}^N$ be a vector of complex numbers. The Discrete Fourier Transform (DFT) takes x as input and gives as output the vector $y = (y_0, y_1, \ldots, y_{N-1}) \in \mathbb{C}^N$ defined componentwise by

$$y_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j \cdot \exp\left(\frac{2\pi i j k}{N}\right), \quad k = 0, 1, \ldots, N-1. \tag{A.1}$$

Here the factor $1/\sqrt{N}$ is a normalization convention used in order to have an unitary matrix associated with such transformation.
The DFT encodes frequency information: if x represents a sampled time-domain signal of period T, then y_k is the amplitude of the k-th frequency mode. From a complexity point of view the DFT can be computed in $\Theta(n \cdot 2^n)$ arithmetic operations, using the most common algorithm known as Fast Fourier Transform (FFT), where $N = 2^n$.

A great discovery in quantum computation has been the translation of this transformation onto a quantum computer.

Definition 8. *Quantum Fourier Transform*
Let $n \in \mathbb{N}$ and set $N = 2^n$. The Quantum Fourier Transform (QFT) is the linear operator on the computational basis $\{|0\rangle, |1\rangle, \ldots, |N-1\rangle\}$ of an n-qubit Hilbert space defined by the action:

$$|j\rangle \longmapsto \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \exp\left(\frac{2\pi i j k}{N}\right) |k\rangle . \tag{A.2}$$

By linearity, if we consider a general state the QFT maps

$$\sum_j x_j \left|j\right\rangle \longmapsto \sum_k y_k \left|k\right\rangle . \tag{A.3}$$

where the amplitudes y_k are computed as the DFT of the amplitudes x_j.

Proposition 4. *The Quantum Fourier Transform is a unitary operator.*

Proof. The transformation is unitary iff the resulting states $|j\rangle$ are such that $\langle j|j'|j|j'\rangle = \delta_{jj'}$

$$\begin{aligned}
\langle j|j'|j|j'\rangle &= \left(\frac{1}{\sqrt{N}}\sum_{k=0}^{N-1}\exp\left(-\frac{2\pi ijk}{N}\right)\langle k|\right)\left(\frac{1}{\sqrt{N}}\sum_{l=0}^{N-1}\exp\left(\frac{2\pi ij'l}{N}\right)|l\rangle\right)\\
&= \frac{1}{N}\sum_{k=0}^{N-1}\sum_{l=0}^{N-1}\exp\left(\frac{2\pi i(j'l-jk)}{N}\right)\langle k|l|k|l\rangle\\
&= \frac{1}{N}\sum_{k=0}^{N-1}\exp\left(\frac{2\pi i(j'-j)k}{N}\right)\\
&= \delta_{jj'}.
\end{aligned}$$

□

A.2 The Product Representation and the associated Circuit

From now on let $N = 2^n$ and consider as usual convention $|0\rangle, |1\rangle, ..., |2^n-1\rangle$ the computational basis for an n-qubit system.
We can write any $|j\rangle \in \{|0\rangle, |1\rangle, ..., |2^n-1\rangle\}$ in binary as $j = j_1 j_2 \dots j_n$, meaning $j = \sum_{\ell=1}^{n} j_\ell \cdot 2^{n-\ell}$ where each $j_\ell \in \{0,1\}$. It is also convinient to adopt the binary fraction notation where $0.j_\ell j_{\ell+1}\dots j_m$ denotes $j_\ell/2 + j_{\ell+1}/4 + \dots + j_m/2^{m-\ell+1}$.

Given such notation we will now introduce a new representation of the QFT which will be useful for the circuit representation of the transformation.

Theorem 3. *In the binary representation $|j_1, \dots, j_n\rangle$, the QFT admits the following tensor-product factorisation:*

$$QFT\,|j\rangle = \frac{(|0\rangle + e^{2\pi i 0.j_n}|1\rangle)\otimes(|0\rangle + e^{2\pi i 0.j_{n-1}j_n}|1\rangle)\otimes\dots\otimes(|0\rangle + e^{2\pi i 0.j_1 j_2\dots j_n}|1\rangle}{2^{n/2}}. \tag{A.4}$$

Proof.

$$\begin{aligned}
\text{QFT}|j\rangle =& \frac{1}{2^{n/2}} \sum_{k=0}^{2^n-1} e^{2\pi ijk/2^n} |k\rangle \\
=& \frac{1}{2^{n/2}} \sum_{k_1=0}^{1} \cdots \sum_{k_n=0}^{1} e^{2\pi ij\left(\sum_{l=1}^{n} k_l 2^{-l}\right)} |k_1 \dots k_n\rangle \\
=& \frac{1}{2^{n/2}} \sum_{k_1=0}^{1} \cdots \sum_{k_n=0}^{1} \bigotimes_{l=1}^{n} e^{2\pi ijk_l 2^{-l}} |k_l\rangle \\
=& \frac{1}{2^{n/2}} \bigotimes_{l=1}^{n} \left[\sum_{k_l=0}^{1} e^{2\pi ijk_l 2^{-l}} |k_l\rangle \right] \\
=& \frac{1}{2^{n/2}} \bigotimes_{l=1}^{n} \left[|0\rangle + e^{2\pi ij2^{-l}} |1\rangle \right] \\
=& \frac{\left(|0\rangle + e^{2\pi i 0.j_n}|1\rangle\right)\left(|0\rangle + e^{2\pi i 0.j_{n-1}j_n}|1\rangle\right)\dots\left(|0\rangle + e^{2\pi i 0.j_1 j_2 \dots j_n}|1\rangle\right)}{2^{n/2}}.
\end{aligned}$$

□

This representation shows that the QFT's output is a tensor product of n single-qubit states, each of which depends on only a suffix of the binary string $(j_1, \dots, j_n)$. Then it immediately suggests a circuit construction.
The circuit introduces a new quantum gate, the controlled R_k gate for $k \geq 1$,which is represented by the unitary matrix:

$$R_k = \begin{pmatrix} 1 & 0 \\ 0 & e^{2\pi i/2^k} \end{pmatrix}. \tag{A.5}$$

Notice that if we multiply j_2 by R_2 we get

$$\begin{pmatrix} 1 & 0 \\ 0 & e^{2\pi i/2^k} \end{pmatrix} \begin{pmatrix} 1 \\ j_2 \end{pmatrix} = \begin{pmatrix} 1 \\ e^{2\pi i 0.0 j_2} \end{pmatrix}.$$

Therefore the QFT on n-qubits can be implemented following the procedure:

1. Apply the Hadamard gate H to the qubit q_1. The state of q_1 becomes

$$(|0\rangle + e^{2\pi i 0.j_1}|1\rangle)/\sqrt{2}$$

.

2. Apply controlled-R_k gates ($k = 2, 3, \dots, n$) to q_1 controlled on $q_2, q_3, \dots, q_n$ respectively, accumulating binary fractional phases to produce

$$(|0\rangle + e^{2\pi i 0.j_1 j_2 \dots j_n}|1\rangle)/\sqrt{2}$$

.

3. Repeat steps 1 and 2 for each subsequent qubit q_m, using only qubits below q_m as controls and applying phases of decreasing depth.

4. Apply SWAP gates to reverse the order of the output qubits, matching the ordering in the above Theorem.

In such way we obtain the following circuit as an output

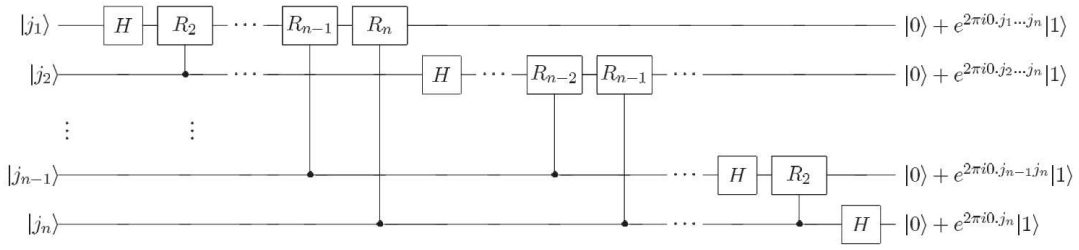


Figure A.1: QFT Circuit

The total gate count is $n + (n-1) + \cdots + 1 = n(n+1)/2$ phase-rotation and Hadamard gates, plus at most $\lfloor n/2 \rfloor$ SWAPs (each decomposable into three CNOT gates). The algorithm therefore runs in $\Theta(n^2)$ elementary gates. So the following result holds.

Corollary 4. *(Complexity of the QFT)*
The QFT on n qubits can be implemented using $\Theta(n^2)$ elementary quantum gates. The best classical algorithm for the DFT on $N = 2^n$ inputs (the FFT) requires $\Theta(n \cdot 2^n)$ operations. Thus the QFT achieves an exponential reduction in gate complexity relative to classical Fourier computation.

It should be emphasised that this exponential speedup does not directly translate to a speedup in computing Fourier Transforms of classical data.
The obstacle is twofold: preparing an arbitrary quantum state encoding classical data requires exponential effort in general, the Fourier coefficients are hidden in the probability amplitude of the state and there is no straightforward way to discover them. Nevertheless, for a certain class of problems it is possible to unleash the power of the quantum Fourier Transform and force an exponential decrease in circuit size.

A.3 Implementation

A direct follow up to this theoretical dissertation about the Quantum Fourier Transform is the implementation in Qiskit. Using the built-in `QFTGate` from the Qiskit library, we can efficiently apply the transformation. The code snippet below illustrates the process and via the decomposition we can have a look on how the built in function is implemented:

```
from qiskit import QuantumCircuit
from qiskit.circuit.library import QFTGate
qc = QuantumCircuit(4)
qc.compose(QFTGate(4), inplace=True)
qc.decompose().draw("mpl")
```

Listing A.1: QFTGate

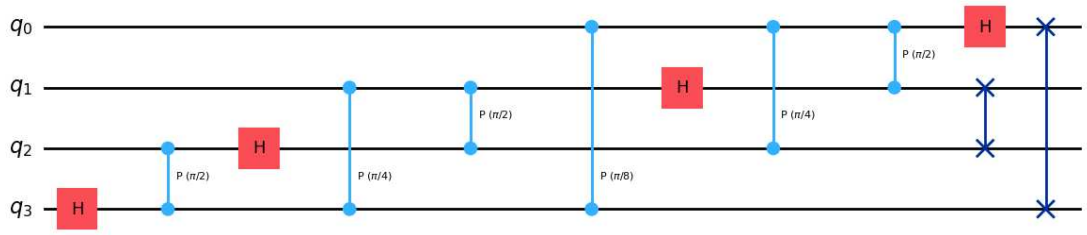


Figure A.2: Output

Bibliography

[1] Dorit Aharonov and Amnon Ta-Shma. *Adiabatic Quantum State Generation and Statistical Zero Knowledge*. 2003. arXiv: quant-ph/0301023 [quant-ph]. URL: https://arxiv.org/abs/quant-ph/0301023.

[2] Dominic W. Berry, Andrew M. Childs, and Robin Kothari. "Hamiltonian Simulation with Nearly Optimal Dependence on all Parameters". In: *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*. IEEE, Oct. 2015, pp. 792–809. DOI: 10.1109/focs.2015.54. URL: http://dx.doi.org/10.1109/FOCS.2015.54.

[3] Dominic W. Berry et al. "Efficient Quantum Algorithms for Simulating Sparse Hamiltonians". In: *Communications in Mathematical Physics* 270.2 (2007), pp. 359–371. DOI: 10.1007/s00220-006-0150-x.

[4] Yudong Cao et al. "Quantum circuit design for solving linear systems of equations". In: *Molecular Physics* 110.15-16 (2012), pp. 1675–1680. DOI: 10.1080/00268976.2012.668289. URL: https://doi.org/10.1080/00268976.2012.668289.

[5] Andrew M. Childs, Robin Kothari, and Rolando D. Somma. "Quantum Algorithm for Systems of Linear Equations with Exponentially Improved Dependence on Precision". In: *SIAM Journal on Computing* 46.6 (Jan. 2017), pp. 1920–1950. ISSN: 1095-7111. DOI: 10.1137/16m1087072. URL: http://dx.doi.org/10.1137/16M1087072.

[6] Andrew M. Childs, Jin-Peng Liu, and Aaron Ostrander. "High-precision quantum algorithms for partial differential equations". In: *Quantum* 5 (Nov. 2021), p. 574. ISSN: 2521-327X. DOI: 10.22331/q-2021-11-10-574. URL: https://doi.org/10.22331/q-2021-11-10-574.

[7] B. D. Clader, B. C. Jacobs, and C. R. Sprouse. "Preconditioned Quantum Linear System Algorithm". In: *Phys. Rev. Lett.* 110 (25 June 2013), p. 250504. DOI: 10.1103/PhysRevLett.110.250504. URL: https://link.aps.org/doi/10.1103/PhysRevLett.110.250504.

[8] Richard P. Feynman. "Simulating physics with computers". In: *International Journal of Theoretical Physics* 21.6 (June 1982), pp. 467–488. DOI: 10.1007/BF02650179. URL: https://doi.org/10.1007/BF02650179.

[9] Aram W. Harrow, Avinatan Hassidim, and Seth Lloyd. "Quantum Algorithm for Linear Systems of Equations". In: *Physical Review Letters* 103.15 (Oct. 2009). ISSN: 1079-7114. DOI: 10.1103/physrevlett.103.150502. URL: http://dx.doi.org/10.1103/PhysRevLett.103.150502.

[10] Junpeng Hu. *Quantum circuits for partial differential equations via schrodingerisation*. https://github.com/hjp3268/Schrodingerisation-Circuits. 2024.

[11] Ali Javadi-Abhari et al. *Quantum computing with Qiskit*. 2024. arXiv: 2405.08810 [quant-ph]. URL: https://arxiv.org/abs/2405.08810.

[12] Shi Jin, Nana Liu, and Yue Yu. "Quantum Simulation of Partial Differential Equations via Schrödingerization". In: *Phys. Rev. Lett.* 133 (23 Dec. 2024), p. 230602. DOI: 10.1103/PhysRevLett.133.230602. URL: https://link.aps.org/doi/10.1103/PhysRevLett.133.230602.

[13] Shi Jin, Nana Liu, and Yue Yu. "Quantum simulation of partial differential equations: Applications and detailed analysis". In: *Phys. Rev. A* 108 (3 Sept. 2023), p. 032603. DOI: 10.1103/PhysRevA.108.032603. URL: https://link.aps.org/doi/10.1103/PhysRevA.108.032603.

[14] Shi Jin et al. "Quantum Circuits for partial differential equations via Schrödingerisation". In: *Quantum* 8 (Dec. 2024), p. 1563. ISSN: 2521-327X. DOI: 10.22331/q-2024-12-12-1563. URL: http://dx.doi.org/10.22331/q-2024-12-12-1563.

[15] Serge Lang. *Algebra*. 3rd ed. Graduate Texts in Mathematics. Springer New York, 2002. ISBN: 978-0-387-95385-4. DOI: 10.1007/978-1-4613-0041-0.

[16] Lin Lin. *Lecture Notes on Quantum Algorithms for Scientific Computation*. 2022. arXiv: 2201.08309 [quant-ph]. URL: https://arxiv.org/abs/2201.08309.

[17] Nana Liu et al. "Power of one qumode for quantum computation". In: *Physical Review A* 93.5 (May 2016). ISSN: 2469-9934. DOI: 10.1103/physreva.93.052304. URL: http://dx.doi.org/10.1103/PhysRevA.93.052304.

[18] Seth Lloyd. "Universal Quantum Simulators". In: *Science* 273.5278 (1996), pp. 1073–1078. DOI: 10.1126/science.273.5278.1073.

[19] Ashley Montanaro and Sam Pallister. "Quantum algorithms and the finite element method". In: *Phys. Rev. A* 93 (3 Mar. 2016), p. 032324. DOI: 10.1103/PhysRevA.93.032324. URL: https://link.aps.org/doi/10.1103/PhysRevA.93.032324.

[20] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. 10th Anniversary Edition. Cambridge: Cambridge University Press, 2010. ISBN: 978-1-107-00217-3.

[21] Yuki Sato et al. "Hamiltonian simulation for hyperbolic partial differential equations by scalable quantum circuits". In: *Phys. Rev. Res.* 6 (3 Sept. 2024), p. 033246. DOI: 10.1103/PhysRevResearch.6.033246. URL: https://link.aps.org/doi/10.1103/PhysRevResearch.6.033246.

[22] Rafaella Vale et al. “Circuit Decomposition of Multicontrolled Special Unitary Single-Qubit Gates”. In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 43.3 (2024), pp. 802–811. DOI: 10.1109/TCAD.2023.3327102.

[23] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, 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.

[24] Shijie Wang et al. “Quantum fast Poisson solver: the algorithm and complete and modular circuit design”. In: *Quantum Information Processing* 19.6 (2020), p. 170. DOI: 10.1007/s11128-020-02669-7.

Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

Versicherung zur selbständigen Leistungserbringung

Titel der Masterarbeit:

Quantum Algorithms for PDEs: Leveraging Schrödingerization

Thema bereitgestellt von (Titel, Vorname, Nachname, Lehrstuhl):
Professor Dr. Christian Klingenberg,
Mathematische Strömungsmechanik

Eingereicht durch (Vorname, Nachname, Matrikel):
Giacomo Lorelli, 3301039

Ich versichere, dass ich die vorstehende schriftliche Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Die benutzte Literatur sowie sonstige Hilfsquellen sind vollständig angegeben. Wörtlich oder dem Sinne nach dem Schrifttum oder dem Internet entnommene Stellen sind unter Angabe der Quelle kenntlich gemacht.

Weitere Personen waren an der geistigen Leistung der vorliegenden Arbeit nicht beteiligt. Insbesondere habe ich nicht die Hilfe eines Ghostwriters oder einer Ghostwriting-Agentur in Anspruch genommen. Dritte haben von mir weder unmittelbar noch mittelbar Geld oder geldwerte Leistungen für Arbeiten erhalten, die im Zusammenhang mit dem Inhalt der vorgelegten Arbeit stehen.

☐ Mit dem Prüfungsleiter bzw. der Prüfungsleiterin wurde abgestimmt, dass für die Erstellung der vorgelegten schriftlichen Arbeit Chatbots (insbesondere ChatGPT) bzw. allgemein solche Programme, die anstelle meiner Person die Aufgabenstellung der Prüfung bzw. Teile derselben bearbeiten könnten, entsprechend den Vorgaben der Prüfungsleiterin bzw. des Prüfungsleiters eingesetzt wurden. Die mittels Chatbots erstellten Passagen sind als solche gekennzeichnet.

Der Durchführung einer elektronischen Plagiatsprüfung stimme ich hiermit zu. Die eingereichte elektronische Fassung der Arbeit ist vollständig. Mir ist bewusst, dass nachträgliche Ergänzungen ausgeschlossen sind.

Die Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht. Ich bin mir bewusst, dass eine unwahre Erklärung zur Versicherung der selbstständigen Leistungserbringung rechtliche Folgen haben kann.

Würzburg, 16/07/2026 Giacomo Lorelli
Ort, Datum, Unterschrift