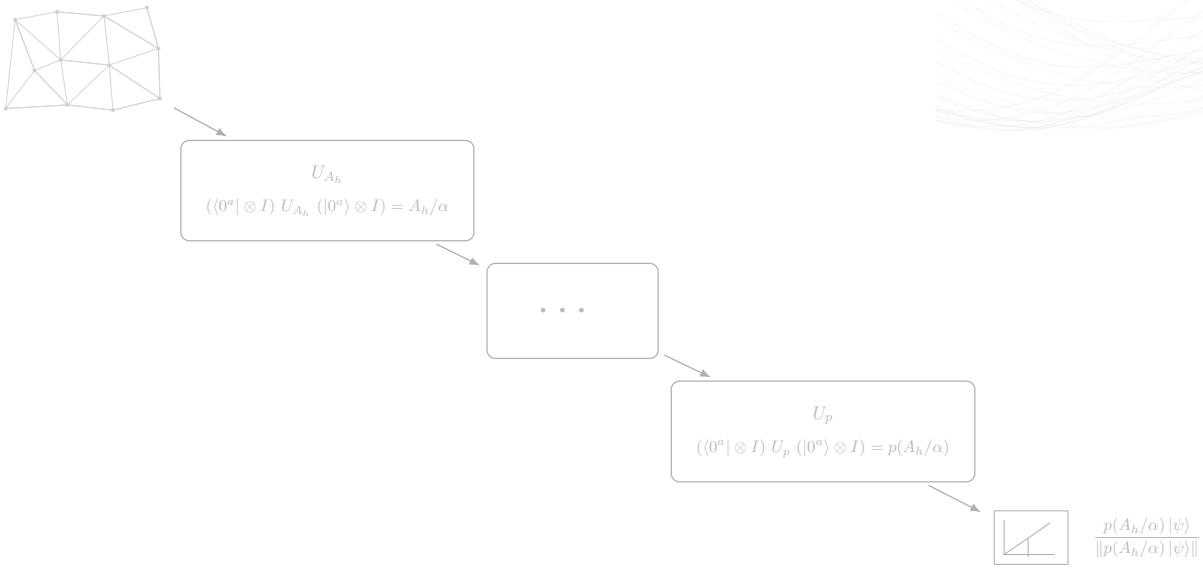

A Quantum Path to Partial Differential Equations

Xiantao Li

July 9, 2026

Contents

Preface

Quantum computing has long been motivated by problems in quantum physics. Over the past two decades, this promise has become increasingly concrete in quantum chemistry and materials science, where Hamiltonian simulation and related algorithms provide a natural connection between physical models and quantum computation. More recently, general scientific computing problems—especially differential equations, including partial differential equations—have emerged as a promising application area outside quantum physics for fault-tolerant quantum algorithms [22, 6].

This book grew out of a series of talks and lectures I have given on this subject. It is written primarily for readers who are familiar with PDEs and their numerical solution and who would like to begin working on the quantum side. It should be equally useful to readers who already have experience with quantum algorithms and would like a numerically grounded introduction to PDE applications, including, specifically, the discretizations behind the matrices, the conditioning, normalization, and measurement questions that determine whether a quantum PDE algorithm is useful in an end-to-end manner.

There are already several excellent introductions to quantum computing and quantum algorithms. Nielsen and Chuang provide the standard comprehensive reference [71]. The lecture notes of Childs and de Wolf give broad and complementary accounts of quantum algorithms and quantum computation [21, 85]. The lecture notes of Lin and Wiebe on quantum algorithms for scientific computation are particularly close in spirit to the present book: they develop block encoding, qubitization, quantum signal processing, quantum singular value transformation, Hamiltonian simulation, and quantum algorithms for linear systems and differential equations from the viewpoint of scientific computation [61].

For readers who learn best by building circuits and running code, Qiskit and IBM Quantum Learning provide an intuitive route from gates and circuits to small algorithmic examples [39, 38]. PennyLane offers a complementary software-oriented entry point, with tutorials illustrating block encodings from matrix access oracles, linear-combination-of-unitaries constructions, and QSVT in practice [80, 5, 81]. More recently, toolboxes have begun to target the block-encoding path directly. For example, Unitaria provides a Python interface for quantum linear algebra through block encodings, allowing users to compose operations such as addition, multiplication, tensor products, and QSVT at the level of matrix-like objects [26]. Qrisp similarly treats block encodings as high-level programming abstractions for quantum linear algebra [74]. These tools are still early, and their interfaces will certainly evolve, but they are an important sign that block encoding is becoming not only a theorem language, but also a programming interface.

The purpose of this book is narrower than these general references and software platforms. Its particular emphasis is on PDEs and on the interaction between quantum algorithms and the

classical numerical structures that arise from their discretization. The hope is not to replace the broader quantum-computing literature, but to give readers a concrete path from finite differences, finite elements, and operator semigroups to quantum primitives, including block encodings, quantum singular value transformation, Hamiltonian simulation, and measurement.

This emphasis is timely. The search for useful quantum applications is now a major theme not only in academic research, but also in industry-facing quantum algorithm development. The recent perspective paper *The Grand Challenge of Quantum Applications*, written by researchers at Google Quantum AI, stresses the need to identify concrete problem instances and to connect abstract algorithmic advantages to real use cases [7]. Differential equations and PDE-like simulation tasks are increasingly visible in this landscape. For example, IBM's Qiskit ecosystem now includes application-oriented PDE solver functions, such as QUICK-PDE developed by ColibriTD [37]; PsiQuantum-affiliated work revisited Carleman-based algorithms for nonlinear dynamics [40]; and Quantinuum-affiliated work develops QFT-based quantum circuits for PDEs in Fourier space [68]. Outside the block-encoding route, PDE-oriented software platforms are also beginning to appear; for instance, SJTU's UnitaryLab emphasizes quantum scientific computing for differential equations through Schrödingerization-based workflows [78]. These examples do not settle the question of practical advantage, but they show that differential equations are becoming a serious meeting point between quantum algorithms, scientific computing, and emerging software stacks.

In seminars and conferences, the question I hear most often from numerical analysts and scientific computing researchers is a practical one: *What are the basic building blocks?* Classical scientific computing has a well-developed answer. Libraries such as BLAS and LAPACK provide a foundation of linear algebraic operations—matrix products, factorizations, linear solves, and eigenvalue routines—on which discretizations, time integrators, and application codes are built. Quantum scientific computing does not yet have a direct counterpart of comparable maturity, but block encoding offers a useful organizing principle.

A block encoding represents a matrix as a distinguished block of a larger unitary operator. Once a discretized differential operator has been placed in this form, a relatively small collection of primitives—quantum singular value transformation, Hamiltonian simulation, linear combinations of unitaries, amplitude amplification, postselection, and measurement—can be composed into algorithms for elliptic, parabolic, and hyperbolic equations. This is not the only possible route to quantum PDE algorithms. It is, however, a flexible and mathematically transparent path, and it allows several different classes of equations to be treated within a common language. This is also the origin of the book's title: it presents *a* quantum path to PDEs, not *the* quantum path. Variational methods, continuous-variable approaches, analog simulation, tensor networks, and problem-specific quantum constructions all deserve attention. The path emphasized here is the block-encoding path because it is one of the clearest ways to see how classical discretizations, matrix functions, and quantum circuits fit together.

At the same time, a PDE is not simply a large linear system waiting to be solved. PDE models come with function spaces, weak formulations, boundary conditions, stability principles, conservation laws, regularity assumptions, discretization errors, and mesh-dependent conditioning. These features are routine in numerical analysis, but they are not always visible when PDE problems are formulated in the quantum computing literature. Conversely, amplitude encoding, block encoding, postselection, quantum signal processing, and quantum singular value transformation are not normally part of the numerical PDE curriculum.

The aim of this book is therefore to build a bridge between the two subjects. Whenever possible, the classical and quantum formulations are placed side by side. The reader can follow the full computational pipeline:

$$\text{continuous PDE} \longrightarrow \text{discretization} \longrightarrow \text{matrix or semidiscrete evolution}$$
$$\longrightarrow \text{quantum encoding} \longrightarrow \text{quantum transformation} \longrightarrow \text{quantity of interest.}$$

The discretization error, quantum algorithmic error, state-preparation cost, success probability, and measurement cost are all parts of this pipeline.

A recurring theme is that a quantum algorithm for a PDE is not merely a faster linear algebra routine. A quantum computer generally prepares a normalized state rather than returning every entry of a solution vector. Consequently, the cost of loading the input, the normalization of the solution, the probability of successful postselection, and the number of measurements needed to extract an observable can be as important as the complexity of the central matrix transformation. An apparent speedup in one stage need not translate automatically into an end-to-end advantage. Keeping these issues visible is essential for a meaningful comparison with classical algorithms.

The organization of the book reflects these goals. Chapter 1 introduces the basic quantum primitives in the language of numerical linear algebra. Readers already familiar with quantum algorithms may skip much of this chapter or use it as a reference; readers coming from numerical PDEs should find its examples—Laplacians, grid functions, and stencils—familiar objects presented in a new vocabulary. Chapters 2, 3, and 4 treat canonical elliptic, hyperbolic, and parabolic model problems. Each chapter begins with familiar finite difference or finite element discretizations and then asks how the resulting matrices or semidiscrete evolutions can be encoded and processed on a quantum computer. Readers from numerical analysis may skim these classical openings and concentrate on the encoding, transformation, and measurement stages. Meanwhile, readers from quantum computing can read the same sections as a compact review of the discrete structures—stiffness and mass matrices, CFL conditions, mesh-dependent conditioning—that any quantum algorithm for PDEs must respect. Chapter 5 gives a brief first look at nonlinear problems through two representative linearization frameworks. The PDE chapters close with outlook sections describing open research directions, and every chapter ends with exercises, so the book can be used for self-study or as the basis of a one-semester topics course.

On the classical PDE side, Evans's book remains a standard and accessible reference for the analytical theory, including weak formulations, energy estimates, regularity, and the function-space viewpoint for elliptic, parabolic, and hyperbolic equations [31]. For readers who already know quantum algorithms and wish to understand the numerical side of PDEs, the books of Larsson and Thomée, Tveito and Winther, and Morton and Mayers provide useful entry points into finite differences, finite elements, stability, and error estimates [53, 84, 69]. Readers familiar with the finite difference and finite element treatments in these books should find the discretizations used here recognizable. On the quantum side, no prior training in quantum mechanics is assumed; elementary linear algebra is the only essential prerequisite. The necessary notation and circuit concepts are introduced as they are needed.

This book is deliberately not a comprehensive survey, and it does not claim that quantum computers will solve PDEs faster than classical computers across the board. Such a claim would be premature and, in many settings, misleading. PDE computation includes nonlinear conservation laws, shocks and singularities, multiscale problems, fluid–structure interaction,

stochastic dynamics, turbulence, moving interfaces, and many other challenges for which the quantum algorithmic picture remains incomplete.

Several important quantum approaches are also treated only briefly or omitted, including variational quantum algorithms, continuous-variable methods, tensor-network approaches, stochastic differential equations, and detailed fault-tolerant resource estimates. Hardware-level compilation, which depends strongly on the native gate set, qubit connectivity, and error-correction architecture of a particular platform, is likewise beyond the scope of this introduction. These omissions are intentional. The objective is to make one coherent route sufficiently explicit that readers can understand its mechanisms, limitations, and possible extensions.

Quantum algorithms for PDEs are still at an early stage. The subject lies at the intersection of quantum algorithms, numerical analysis, applied mathematics, and scientific computing, and substantial progress will require ideas from all of these areas. My hope is that this book provides a useful entry point: a first map for students, a common vocabulary for researchers from different communities, and an invitation to develop sharper algorithms, stronger analysis, and more realistic applications.

If the book helps a few more researchers cross the boundary between numerical PDEs and quantum computation, it will have served its purpose. The author gratefully acknowledges the support of the National Science Foundation under Grants No. DMS-2411120 and DMS-2552687. Suggestions, corrections, and questions are welcome and may be sent to Xiantao.Li@psu.edu.

Xiantao Li
Department of Mathematics, Pennsylvania State University
University Park, Pennsylvania

Chapter 1

Basic Quantum Elements for PDE Algorithms

1.1 An overview

This chapter introduces a small collection of quantum primitives that will be used throughout this lecture note. The intended reader is familiar with finite-dimensional linear algebra, matrix factorizations, finite differences, finite elements, and operator discretization, but is not assumed to know quantum mechanics.

The central message is simple. A quantum computer stores a vector in an exponentially large Hilbert space, but only through unitary transformations and measurements. Thus the relevant question is not merely whether a vector or a matrix can be represented, but whether it can be represented in a form that can be manipulated, transformed, and measured efficiently. The primitive that will play the role of a matrix access model is the *block encoding*. Once an operator has been block encoded, the quantum singular value transformation (QSVT) gives a systematic way to apply polynomial functions to that operator. This is the quantum analogue of a familiar numerical-analysis idea: approximate a function of a matrix by a polynomial, then apply the resulting polynomial through repeated matrix-vector-like operations. Standard background on the circuit model and the postulates of quantum mechanics can be found in Nielsen and Chuang [71]; Lin and Wiebe's lecture notes provide a complementary introduction oriented toward scientific computing and matrix algorithms [60, 61].

The chapter is organized as follows. We first introduce quantum states, computational-basis vectors, Pauli strings, tensor-product registers, and a short norm dictionary, followed by measurement, grid-function amplitude encoding, and state preparation. The transform-and-estimate primitives come next: the quantum Fourier transform, quantum phase estimation, inner-product estimation, postselection, amplitude amplification, and amplitude estimation. We then develop the matrix access model: block encodings and their calculus, diagonal and sparse-matrix access, QSVT, and explicit block encodings of elementary finite-difference operators. A short section connects these primitives to PDE algorithms, illustrated by two worked examples—a Poisson solver and a Schrödinger evolution built on the same discrete Laplacian—then assembles the pieces, and a final section explains how to read query and gate-count estimates.

The notation introduced here is used consistently in the PDE chapters. The amplitude

encoding of grid vectors in (1.33) and the L^2–ℓ_2 scaling in (1.35) are the bridge between functions and quantum states. The QFT definition in (1.48) is the structured-grid spectral transform used later when periodic or tensor-product boundary conditions allow diagonalization. The block-encoding definition (1.83) is the matrix access model used in the elliptic, hyperbolic, and parabolic chapters. The finite-difference block encodings in Section 1.13, especially (1.133) and (1.140), should be viewed as the first concrete examples of the operator encodings used later in Chapters 2–4. Throughout the lecture note, $\widetilde{O}(\cdot)$ suppresses logarithmic factors in dimension, precision, condition-number, and related parameters.

Remark 1.1. *The presentation deliberately avoids physical language when possible. The word "state" should be read as "unit-norm vector in a complex vector space." The word "gate" should be read as "unitary matrix with a compact circuit representation." The word "measurement" should be read as "randomized access to the squared moduli of coordinates in a specified basis."*

1.2 Quantum states as normalized vectors

Let n be a positive integer and set $N = 2^n$. An n-qubit pure quantum state is a vector [71, 61]

$$|\psi\rangle = \sum_{j=0}^{N-1} \psi_j \, |j\rangle \in \mathbb{C}^N, \qquad \sum_{j=0}^{N-1} |\psi_j|^2 = 1. \tag{1.1}$$

The vectors $|0\rangle, \ldots, |N-1\rangle$ naturally form the *computational basis*. They are ordinary Euclidean basis vectors, but indexed by binary strings. If the integer j has the binary representation

$$j = (j_{n-1}, \ldots, j_1, j_0)_2 := j_{n-1} 2^{n-1} + \cdots + j_1 2 + j_0, \qquad j_k \in \{0, 1\}, \tag{1.2}$$

where the subscript 2 denotes base two, then

$$|j\rangle = |j_{n-1}\rangle \otimes \cdots \otimes |j_1\rangle \otimes |j_0\rangle \, . \tag{1.3}$$

The two basis states of one qubit are

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \qquad |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}. \tag{1.4}$$

Thus a quantum state is not mysterious: it is a unit vector. The difference from classical numerical linear algebra is operational. One does not read the vector entries directly. A measurement of $|\psi\rangle$ in the computational basis returns the integer j with probability $|\psi_j|^2$.

1.2.1 Gates as unitary matrices

A quantum circuit typically applies a product of unitary matrices to a state. Common one-qubit gates include [71, 61]

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \qquad Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, \qquad H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}. \tag{1.5}$$

It is often helpful to look at what a gate operation does to some specific states. For example, the Hadamard gate satisfies

$$H|0\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}}, \qquad H|1\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}}. \tag{1.6}$$

This can be generalized to multiple qubits. Applying H to every qubit gives the uniform superposition

$$H^{\otimes n}|0^n\rangle = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} |j\rangle . \tag{1.7}$$

This is the quantum analogue of initializing all grid points at once.

1.2.2 Pauli matrices and Pauli strings

For quantum algorithms based on Hamiltonian simulation, it is useful to have a standard basis for matrices on qubits. The one-qubit Pauli matrices are [71]

$$I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \qquad X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \qquad Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}, \qquad Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}. \tag{1.8}$$

They are Hermitian and unitary, and satisfy

$$X^2 = Y^2 = Z^2 = I, \qquad XY = iZ, \qquad YZ = iX, \qquad ZX = iY. \tag{1.9}$$

Consequently,

$$[X, Y] = 2iZ, \qquad [Y, Z] = 2iX, \qquad [Z, X] = 2iY, \tag{1.10}$$

and the nonidentity Pauli matrices X, Y, Z pairwise anticommute, equivalently

$$\{X, Y\} = \{Y, Z\} = \{Z, X\} = 0. \tag{1.11}$$

A Pauli string on n qubits is a tensor product

$$P_\mu = \sigma_{\mu_1} \otimes \cdots \otimes \sigma_{\mu_n}, \qquad \sigma_{\mu_j} \in \{I, X, Y, Z\}. \tag{1.12}$$

Every Pauli string is Hermitian and unitary. Two Pauli strings either commute or anticommute. More precisely, their product differs from the reverse product by a sign determined by the number of tensor positions at which the nonidentity factors anticommute.

The 4^n Pauli strings form an orthogonal basis for all $2^n \times 2^n$ complex matrices with respect to the Hilbert–Schmidt inner product:

$$\mathrm{Tr}(P_\mu^\dagger P_\nu) = 2^n \delta_{\mu\nu}. \tag{1.13}$$

Thus any matrix $H \in \mathbb{C}^{2^n \times 2^n}$ can be expanded as

$$H = \sum_{\mu \in \{I,X,Y,Z\}^n} h_\mu P_\mu, \qquad h_\mu = 2^{-n} \mathrm{Tr}(P_\mu H). \tag{1.14}$$

If H is Hermitian, then all coefficients h_μ are real. This expansion is conceptually simple, but it is not automatically efficient: a sparse matrix, such as the tridiagonal Laplacian, need not be sparse in the Pauli basis.

Remark 1.2 (Commutators and product formulas)**.** *Pauli commutation relations are the algebraic source of many Hamiltonian-simulation estimates. A unitary evolution of a Pauli string can be implemented directly on the quantum circuit [71]. If $H = A + B$ has two non-commuting Pauli strings, a first-order product formula replaces e^{-itH} by $e^{-itA}e^{-itB}$, and the leading error is controlled by commutators such as $[A, B]$. When A and B are sums of Pauli strings, their commutators can be evaluated from the pairwise Pauli commutation rules above.*

1.2.3 Tensor products and register notation

If $|\phi\rangle \in \mathbb{C}^{N_1}$ and $|\psi\rangle \in \mathbb{C}^{N_2}$, then their tensor product is a vector in $\mathbb{C}^{N_1 N_2}$:

$$|\phi\rangle\,|\psi\rangle = |\phi\rangle \otimes |\psi\rangle\,. \tag{1.15}$$

For readability we will usually omit the tensor symbol. For example,

$$|0\rangle\,|j\rangle = |0\rangle \otimes |j\rangle\,. \tag{1.16}$$

A group of qubits is called a *register*. In numerical terms, a register is simply a tensor factor in the ambient vector space [71, 61].

A recurring quantum-algorithmic construction is to enlarge the space containing a system state $|\psi\rangle \in \mathcal{H}_{\mathrm{s}}$ by appending an ancilla register initialized in a known state, usually $|0^a\rangle \in \mathcal{H}_{\mathrm{a}}$. One then applies a joint unitary

$$U_{\mathrm{joint}} : \mathcal{H}_{\mathrm{a}} \otimes \mathcal{H}_{\mathrm{s}} \longrightarrow \mathcal{H}_{\mathrm{a}} \otimes \mathcal{H}_{\mathrm{s}}. \tag{1.17}$$

The additional register may store a branch label, a success flag, an eigenvalue estimate, or temporary arithmetic. Measuring or projecting the ancilla at the end extracts information about the system block. Block encodings, phase estimation, the Hadamard test, and amplitude amplification are all instances of this general "enlarge–evolve–read out" pattern.

Figure 1.1: A system register is elevated to the larger space $\mathcal{H}_{\mathrm{a}} \otimes \mathcal{H}_{\mathrm{s}}$ by appending an ancilla register, followed by a unitary that acts jointly on both registers. Later circuits specialize the joint unitary and the final ancilla readout.

1.3 A short norm dictionary

Before introducing block encodings, we record several matrix norms that appear repeatedly in complexity estimates. The operator norm, or spectral norm, is

$$\|A\|_2 = \max_{\boldsymbol{x}\neq 0} \frac{\|A\boldsymbol{x}\|_2}{\|\boldsymbol{x}\|_2}. \tag{1.18}$$

For Hermitian A, this is the largest absolute eigenvalue. For a diagonal matrix D with entries $d_0, \ldots, d_{N-1}$,

$$\|D\|_2 = \max_j |d_j|. \tag{1.19}$$

The entrywise maximum norm is also commonly used,

$$\|A\|_{\max} = \max_{i,j} |A_{ij}|. \tag{1.20}$$

It is not equivalent to the spectral norm with dimension-independent constants. For an $N \times N$ matrix,

$$\|A\|_{\max} \leq \|A\|_2 \leq N \|A\|_{\max}. \tag{1.21}$$

The first inequality follows by testing on coordinate vectors; the second follows from the Frobenius norm. A sharper and often more useful estimate is

$$\|A\|_2 \leq \sqrt{\|A\|_1 \|A\|_\infty}, \tag{1.22}$$

where $\|A\|_1$ is the maximum column sum and $\|A\|_\infty$ is the maximum row sum. In particular, if A has at most s nonzero entries in each row and column, then

$$\|A\|_2 \leq s \|A\|_{\max}. \tag{1.23}$$

These elementary inequalities matter because a block encoding represents A/α, not A itself. The normalization α must be at least of the order of a norm bound for A. A crude bound, such as $N \|A\|_{\max}$ for a sparse operator, may be much worse than the natural sparse bound $s \|A\|_{\max}$ or the true spectral norm. In PDE discretizations, tracking this normalization is as important as tracking the condition number.

Remark 1.3 (Soft-O notation)**.** *We write $\widetilde{O}(\cdot)$ to suppress factors that are logarithmic in the dimension, precision, condition number, or other explicitly named parameters. This convention is common in quantum algorithms, but in PDE applications the hidden logarithms and state-preparation costs may still be important.*

1.4 Measurement and observables

A measurement is the main point at which quantum notation differs from standard matrix computation. A circuit may prepare or transform a vector in a high-dimensional space, but the final readout is random. The projective-measurement rule used below is one of the basic postulates of quantum mechanics [71, Sec. 2.2]. The simplest case is measurement in the computational basis: for

$$|\psi\rangle = \sum_{j=0}^{N-1} \psi_j |j\rangle, \tag{1.24}$$

the observed index is j with probability $|\psi_j|^2$.

More generally, a directly measurable scalar quantity is represented by a Hermitian matrix, called an *observable*. Let

$$O = \sum_r \lambda_r \Pi_r \tag{1.25}$$

be its spectral decomposition, where the eigenvalues λ_r are distinct and Π_r is the orthogonal projector onto the corresponding eigenspace. Measuring O in the state $|\psi\rangle$ returns the eigenvalue λ_r with probability

$$p_r = \|\Pi_r |\psi\rangle\|_2^2 = \langle\psi| \Pi_r |\psi\rangle . \tag{1.26}$$

If $p_r > 0$, the post-measurement state conditioned on observing λ_r is

$$\frac{\Pi_r |\psi\rangle}{\sqrt{p_r}}. \tag{1.27}$$

Thus direct measurement samples from the spectral measure of O induced by $|\psi\rangle$; it does not return the expectation value in one shot. The expectation is recovered only statistically:

$$\mathbb{E}[\lambda] = \sum_r \lambda_r p_r = \langle\psi| O |\psi\rangle . \tag{1.28}$$

This is often the most useful numerical interpretation: a quantum measurement gives samples of eigenvalues of the observable, with weights determined by the squared projected components of the state.

Example 1.4 (Computational-basis measurement as an observable)**.** *Computational-basis measurement is the special case in which*

$$O = \sum_{j=0}^{N-1} j |j\rangle \langle j| . \tag{1.29}$$

Then $\Pi_j = |j\rangle \langle j|$ *and* (1.26) *gives* $p_j = |\psi_j|^2$.

Example 1.5 (Diagonal observables for grid quantities)**.** *Let* $q(x)$ *be a real-valued grid observable and define*

$$O_q = \sum_{j=0}^{N-1} q(x_j) |j\rangle \langle j| . \tag{1.30}$$

Let $|\boldsymbol{f}_h\rangle \propto (f(x_0), \ldots, f(x_{N-1}))^T$ *be a function defined on some grid points. Measuring* O_q *on* $|\boldsymbol{f}_h\rangle$ *samples the values* $q(x_j)$ *with probabilities proportional to* $|f(x_j)|^2$*. Its expectation is*

$$\langle \boldsymbol{f}_h| O_q |\boldsymbol{f}_h\rangle = \frac{\sum_j q(x_j)|f(x_j)|^2}{\sum_j |f(x_j)|^2}, \tag{1.31}$$

which is a normalized weighted average. To recover the corresponding unnormalized continuum integral, one must multiply by the appropriate norm and quadrature factors.

Remark 1.6 (Expectation estimation is a sampling problem)**.** *If the eigenvalues of* O *lie in* $[a, b]$*, then estimating* $\langle\psi| O |\psi\rangle$ *by repeated direct measurements has Monte Carlo error proportional to* $(b - a)/\sqrt{M}$ *after* M *samples, up to constants depending on the variance. More elaborate procedures, such as amplitude estimation or Hadamard-test-based methods, may improve the scaling under additional circuit assumptions.*

1.5 Grid functions, amplitude encoding, and the L^2–ℓ_2 scaling

Quantum algorithms for PDEs usually begin after spatial discretization. Suppose $\Omega \subset \mathbb{R}^d$ is discretized by N grid points x_j, $j = 0, \dots, N-1$. For simplicity assume a uniform mesh with volume element h^d. A function f is represented by the grid vector

$$\boldsymbol{f}_h = (f(x_0), \dots, f(x_{N-1}))^T \in \mathbb{C}^N. \tag{1.32}$$

A natural quantum representation of this grid vector is the amplitude encoding

$$|\boldsymbol{f}_h\rangle = \frac{1}{\|\boldsymbol{f}_h\|_2} \sum_{j=0}^{N-1} f(x_j)\,|j\rangle\,, \qquad \|\boldsymbol{f}_h\|_2 = \left(\sum_{j=0}^{N-1} |f(x_j)|^2\right)^{1/2}. \tag{1.33}$$

This encoding is homogeneous: $\boldsymbol{f}_h$ and $c\boldsymbol{f}_h$ define the same quantum state up to a global phase when $c \neq 0$. Therefore the scale $\|\boldsymbol{f}_h\|_2$ must be stored or estimated separately if physical units or absolute magnitudes are needed.

1.5.1 The norm conversion

The continuum norm is approximated by the quadrature formula

$$\|f\|^2_{L^2(\Omega)} \approx h^d \sum_{j=0}^{N-1} |f(x_j)|^2 = h^d \,\|\boldsymbol{f}_h\|_2^2\,. \tag{1.34}$$

Hence

$$\|f\|_{L^2(\Omega)} \approx h^{d/2}\,\|\boldsymbol{f}_h\|_2\,. \tag{1.35}$$

Equivalently, if $|\boldsymbol{f}_h\rangle$ is the normalized amplitude encoding in (1.33), then its jth amplitude is approximately

$$\frac{f(x_j)}{\|\boldsymbol{f}_h\|_2} \approx \frac{h^{d/2} f(x_j)}{\|f\|_{L^2(\Omega)}}. \tag{1.36}$$

Thus amplitude encoding stores the L^2-normalized function values multiplied by the square root of the cell volume. This is the same scaling that appears when finite element coefficient vectors are related to L^2 inner products through a mass matrix; see, for example, standard finite-difference and finite-element treatments in [53, 84].

Remark 1.7 (Mass matrices)**.** *For a finite element basis $\{\varphi_j\}_{j=0}^{N-1}$, a coefficient vector $\boldsymbol{u}$ for a function $u(x) = \sum_j u_j \varphi_j(x)$ satisfies*

$$\|u_h\|^2_{L^2} = \boldsymbol{u}^\dagger M \boldsymbol{u}, \qquad M_{ij} = \int_\Omega \overline{\varphi_i(x)}\varphi_j(x)\,dx. \tag{1.37}$$

For real-valued finite element bases the conjugation is invisible, but the Hermitian form above is the convention used in the quantum chapters. Amplitude encoding directly normalizes $\boldsymbol{u}$ in the Euclidean norm. If $M \neq cI$, then the quantum state $|\boldsymbol{u}\rangle = \boldsymbol{u}/\|\boldsymbol{u}\|_2$ is not automatically an L^2-normalized finite element function. One must either use a mass-lumping, apply a transformation involving $M^{1/2}$, or account for M as part of the observable.

1.5.2 Inner products of functions

If f and g are represented by amplitude states $|\boldsymbol{f}_h\rangle$ and $|\boldsymbol{g}_h\rangle$, then

$$\langle \boldsymbol{f}_h | \boldsymbol{g}_h \rangle = \frac{\boldsymbol{f}_h^\dagger \boldsymbol{g}_h}{\|\boldsymbol{f}_h\|_2 \, \|\boldsymbol{g}_h\|_2}. \tag{1.38}$$

The continuum inner product is approximated by

$$\langle f, g \rangle_{L^2} \approx h^d \boldsymbol{f}_h^\dagger \boldsymbol{g}_h = \left(h^{d/2} \, \|\boldsymbol{f}_h\|_2 \right) \left(h^{d/2} \, \|\boldsymbol{g}_h\|_2 \right) \langle \boldsymbol{f}_h | \boldsymbol{g}_h \rangle \, . \tag{1.39}$$

Equation (1.39) is often the first piece of bookkeeping needed in a PDE application. The quantum circuit estimates the normalized overlap; the numerical analyst must multiply back the discrete norms and quadrature weights. This is the same bookkeeping that underlies finite-difference and finite-element error estimates in standard numerical PDE texts such as Larsson–Thomée and Tveito–Winther [53, 84].

1.6 State preparation

A state preparation circuit for a vector $\boldsymbol{v} \in \mathbb{C}^N$ is a unitary U_v such that

$$U_{\boldsymbol{v}} \, |0^n\rangle = |\boldsymbol{v}\rangle := \frac{1}{\|\boldsymbol{v}\|_2} \sum_{j=0}^{N-1} v_j \, |j\rangle \, . \tag{1.40}$$

State preparation can be regarded as a data-input problem. Quantum speedups can disappear if preparing $|\boldsymbol{v}\rangle$ costs $\Theta(N)$. For PDEs, we therefore distinguish between at least three cases:

(i) $\boldsymbol{v}$ has a compact formula, such as samples of a smooth function;

(ii) $\boldsymbol{v}$ is generated by a previous quantum subroutine;

(iii) $\boldsymbol{v}$ is arbitrary classical data and must be loaded from memory.

Only the first two cases usually support efficient state preparation without a strong memory-access assumption.

1.6.1 Grover–Rudolph preparation for smooth densities

Grover and Rudolph introduced an efficient recursive procedure for preparing states associated with efficiently integrable probability distributions [34]. We describe it in a form useful for grid functions.

Let $\rho : [0,1] \to \mathbb{R}_+$ be a probability density and assume that integrals of ρ over dyadic intervals can be computed efficiently. Specifically, let $N = 2^n$ and define cells

$$I_j = [j/N, (j+1)/N), \qquad p_j = \int_{I_j} \rho(x) \, dx. \tag{1.41}$$

The target state is

$$|p\rangle = \sum_{j=0}^{N-1} \sqrt{p_j}\,|j\rangle\,. \tag{1.42}$$

which is automatically normalized. The construction is a binary tree (Figure 1.2). At each node, one rotates the next qubit according to the relative mass in the two children of the current interval.

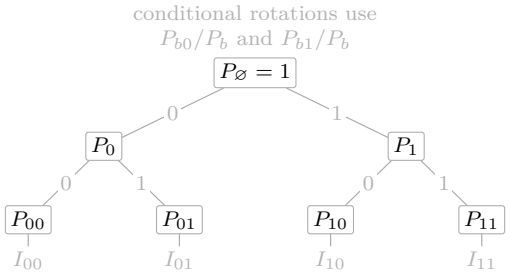

Figure 1.2: The binary tree behind Grover–Rudolph state preparation. The node weight P_b is the integral of the density over the dyadic interval indexed by the bit string b. The rotation at node b places the next qubit in a superposition with probabilities P_{b0}/P_b and P_{b1}/P_b.

Due to the dyadic partitioning, the interval I_b indexed by a bit string $b = b_1 \cdots b_m$, the strings $b0$ and $b1$ are obtained by appending one additional binary digit. The corresponding intervals I_{b0} and I_{b1} are respectively the left and right halves of I_b. Define

$$P_b = \int_{I_b} \rho(x)\,dx, \qquad P_{b0} = \int_{I_{b0}} \rho(x)\,dx, \qquad P_{b1} = \int_{I_{b1}} \rho(x)\,dx. \tag{1.43}$$

The additivity identity $P_b = P_{b0} + P_{b1}$ is the local mass conservation property of the binary tree. The controlled rotation at node b uses the angle θ_b satisfying

$$\cos^2\theta_b = \frac{P_{b0}}{P_b}, \qquad \sin^2\theta_b = \frac{P_{b1}}{P_b}. \tag{1.44}$$

After all n levels, the amplitude of the leaf indexed by $j = j_1 \cdots j_n$ is[1]

$$\prod_{m=0}^{n-1} \sqrt{\frac{P_{j_1\cdots j_{m+1}}}{P_{j_1\cdots j_m}}} = \sqrt{P_j} = \sqrt{p_j}, \tag{1.45}$$

where the empty string is used at $m = 0$. Thus the telescoping product of conditional probabilities gives the target amplitudes.

Remark 1.8 (Preparing a function, not a density)**.** *If the desired amplitudes are proportional to a real-valued function f, set*

$$\rho(x) = \frac{|f(x)|^2}{\int |f(y)|^2\,dy}. \tag{1.46}$$

[1]Here bits are generated from the most significant downward, so j_1 is the most significant bit; in the register convention $|j\rangle = |j_{n-1}\rangle \otimes \cdots \otimes |j_0\rangle$ introduced earlier this is the bit j_{n-1}.

Grover–Rudolph preparation produces amplitudes proportional to $|f|$. The sign or complex phase of f must then be inserted by a phase oracle:

$$|j\rangle \mapsto e^{i \arg f(x_j)} |j\rangle . \tag{1.47}$$

For sign-changing real functions this is a ± 1 phase. For smooth complex functions it requires an efficient approximation of the phase.

Remark 1.9 (Higher dimensions)**.** *For $\Omega = [0,1]^d$, one may recursively bisect boxes or use a tensor-product construction when $\rho(x_1, \ldots, x_d)$ is separable or approximately separable. The computational bottleneck is the ability to evaluate integrals over dyadic boxes. This is a useful point of contact with adaptive quadrature and hierarchical bases in numerical analysis.*

An alternative, more general state-preparation network was given by Kaye and Mosca [50]. Their construction also uses conditional norms and phases associated with a binary decomposition. It is efficient for structured families for which these quantities can be computed efficiently, but it does not make arbitrary classical data inexpensive to load.

1.7 The quantum Fourier transform

The quantum Fourier transform (QFT) is the unitary version of the discrete Fourier transform acting on amplitudes. Let $N = 2^n$ and let $\omega_N = e^{2\pi i/N}$. The QFT is defined by the following unitary; see Nielsen and Chuang [71, Sec. 5.1] for detailed derivations:

$$F_N |j\rangle = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \omega_N^{jk} |k\rangle , \qquad j = 0, \ldots, N-1. \tag{1.48}$$

The inverse QFT is $F_N^\dagger$, obtained by replacing ω_N by ω_N^{-1}:

$$F_N^\dagger |k\rangle = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} \omega_N^{-jk} |j\rangle . \tag{1.49}$$

If a grid vector is amplitude encoded as $|\boldsymbol{u}\rangle = \sum_j u_j |j\rangle / \|u\|$, then applying F_N produces the normalized vector of discrete Fourier coefficients. This is why QFT diagonalizes periodic finite-difference operators in later chapters.

The circuit follows from the binary factorization of a Fourier basis state. With the binary fraction

$$0.j_\ell j_{\ell-1} \cdots j_0 := \frac{j_\ell}{2} + \frac{j_{\ell-1}}{2^2} + \cdots + \frac{j_0}{2^{\ell+1}}, \tag{1.50}$$

one obtains, up to reversal of the output-qubit order,

$$F_N |j_{n-1} \cdots j_0\rangle = \frac{1}{2^{n/2}} \bigotimes_{\ell=0}^{n-1} \left(|0\rangle + e^{2\pi i (0.j_\ell \cdots j_0)} |1\rangle \right). \tag{1.51}$$

Each factor is generated by one Hadamard gate followed by controlled phase rotations; the final SWAP gates restore the conventional bit order; Figure 1.3 shows the four-qubit circuit.

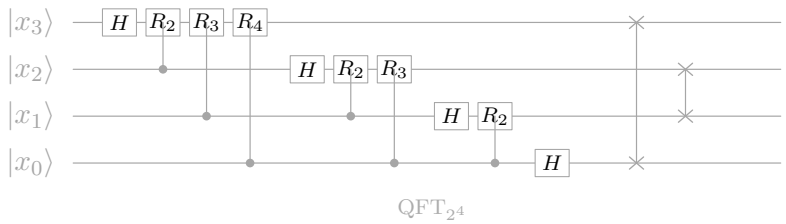

Figure 1.3: A QFT circuit on four qubits. Here $R_m = \mathrm{diag}(1, e^{2\pi i/2^m})$. The final swaps implement bit reversal. The inverse QFT reverses the gate order and replaces each R_m by $R_m^\dagger$.

The classical fast Fourier transform maps an explicitly stored vector in $\mathbb{C}^N$ to another explicitly stored vector in $O(N \log N)$ arithmetic operations. The QFT instead maps an amplitude-encoded vector on $n = \log_2 N$ qubits to another amplitude-encoded vector using $O(n^2)$ one- and two-qubit gates for the exact circuit, and fewer gates for standard approximate versions [23, 71]. The apparent exponential gap should be interpreted carefully: the QFT does not print all Fourier coefficients. It prepares a state whose amplitudes are those coefficients, and subsequent measurements or controlled operations can access only selected information.

The circuit structure is nevertheless important. Rather than updating N Fourier coefficients one at a time, the gates act coherently on all amplitudes of the quantum state. The controlled phase gates are diagonal and many commute, while gates on disjoint qubit pairs can be scheduled in parallel. Hence the exact circuit uses $O(n^2)$ gates, and its depth can be reduced further subject to connectivity and approximation tolerance. The saving is therefore a combination of coherent amplitude processing and circuit parallelism, not the classical production of N output numbers. This is one of the reasons QFT-based methods are attractive for structured grids, where the PDE operator is diagonal or nearly diagonal in the Fourier basis.

Remark 1.10 (QFT versus FFT)**.** *For numerical analysts, the safest analogy is the following. The FFT is an efficient explicit change of basis for a stored vector. The QFT is an efficient coherent change of basis for an amplitude-encoded vector. It is extremely powerful when a quantum algorithm needs to apply phases, filters, or measurements in the frequency basis; it is not a replacement for the FFT when the task is to output the full list of Fourier coefficients as classical data.*

1.8 Quantum phase estimation

Quantum phase estimation (QPE) is the standard primitive for converting an eigenphase into a binary register. It is the main reason the inverse QFT was introduced in the previous section. Suppose that a unitary U satisfies

$$U |q_j\rangle = e^{2\pi i \theta_j} |q_j\rangle , \qquad 0 \le \theta_j < 1. \tag{1.52}$$

QPE uses an r-qubit clock register, controlled powers $U, U^2, U^4, \ldots, U^{2^{r-1}}$, and an inverse QFT on the clock register. On an exact eigenstate, the idealized map is

$$|0^r\rangle |q_j\rangle \longmapsto \left|\widetilde{\theta}_j\right\rangle |q_j\rangle , \tag{1.53}$$

where $\widetilde{\theta}_j$ is an r-bit approximation of θ_j. More precisely, r clock qubits resolve phases to accuracy $O(2^{-r})$, with the usual constant-success-probability qualification; additional qubits or median amplification improve the success probability [71, Sec. 5.2].

The real power of QPE is that it acts coherently on superpositions. If

$$|\psi\rangle = \sum_j c_j \, |q_j\rangle \,, \tag{1.54}$$

then QPE produces

$$|0^r\rangle \, |\psi\rangle \longmapsto \sum_j c_j \left|\widetilde{\theta}_j\right\rangle |q_j\rangle \,, \tag{1.55}$$

up to the phase-estimation error. Thus QPE does not merely return one eigenvalue; it attaches approximate spectral information to every eigenmode present in the input state. This viewpoint is central to eigenvalue problems, linear-system algorithms, and amplitude estimation. The latter applies QPE to the two-dimensional Grover rotation introduced in Subsection 1.10.4. In Chapter 2, we will apply the same idea to elliptic eigenvalue problems.

For Hamiltonian simulation, one solves the Schrödinger equation,

$$U = e^{-iHt_0}. \tag{1.56}$$

If $H \, |q_j\rangle = \lambda_j \, |q_j\rangle$, then the measured phase is $\theta_j = -t_0\lambda_j/(2\pi)$ modulo one. The modulo-one ambiguity is important: either t_0 must be small enough to avoid aliasing on the spectral window of interest, or additional information must be used to unwrap the phase. This mesh-dependent aliasing issue will appear again for discretized PDE operators, whose largest eigenvalues often grow like a power of h^{-1}.

More advanced variants, including iterative, adaptive, and fixed-point phase estimation, reduce ancilla counts or improve robustness. We will not need those refinements in this introductory chapter, but they are important in resource-sensitive implementations.

1.9 Estimating inner products

Many quantities of interest in numerical PDEs are inner products: masses, energies, weak-form residuals, correlations, and linear functionals. Quantum circuits provide several ways to estimate such quantities. We record the two most common primitives.

1.9.1 The swap test

Given two states $|\boldsymbol{u}\rangle$ and $|\boldsymbol{v}\rangle$, the swap test estimates $|\langle \boldsymbol{u}|\boldsymbol{v}\rangle|^2$. It uses one ancilla qubit and a controlled-SWAP operation (Figure 1.4). It can also be viewed as a Hadamard test whose controlled unitary is the register-swap operator. The test appears naturally in quantum fingerprinting and is a standard primitive for estimating fidelities and overlaps [18, 71, 61].

We will show a sequence of states to make the construction transparent. We attach an ancilla register and starting from

$$|\Psi_0\rangle = |0\rangle \, |\boldsymbol{u}\rangle \, |\boldsymbol{v}\rangle \,, \tag{1.57}$$

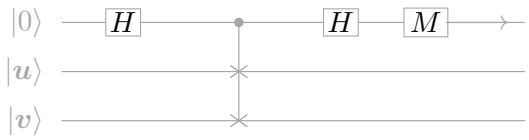

Figure 1.4: Swap test for estimating $|\langle \boldsymbol{u}|\boldsymbol{v}\rangle|^2$. The controlled operation swaps the two input registers only when the ancilla is $|1\rangle$, and the box M denotes computational-basis measurement of the ancilla.

the first Hadamard gate gives

$$|\Psi_1\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle|\boldsymbol{u}\rangle|\boldsymbol{v}\rangle + |1\rangle|\boldsymbol{u}\rangle|\boldsymbol{v}\rangle\right). \tag{1.58}$$

The controlled-SWAP acts only on the second branch,

$$|\Psi_2\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle|\boldsymbol{u}\rangle|\boldsymbol{v}\rangle + |1\rangle|\boldsymbol{v}\rangle|\boldsymbol{u}\rangle\right). \tag{1.59}$$

After the second Hadamard gate on the ancilla,

$$\begin{aligned}|\Psi_3\rangle = {} & \frac{1}{2}|0\rangle\left(|\boldsymbol{u}\rangle|\boldsymbol{v}\rangle + |\boldsymbol{v}\rangle|\boldsymbol{u}\rangle\right) \\ & + \frac{1}{2}|1\rangle\left(|\boldsymbol{u}\rangle|\boldsymbol{v}\rangle - |\boldsymbol{v}\rangle|\boldsymbol{u}\rangle\right).\end{aligned} \tag{1.60}$$

Therefore

$$\begin{aligned}\Pr(\text{ancilla} = 0) &= \frac{1}{4}\left\||\boldsymbol{u}\rangle|\boldsymbol{v}\rangle + |\boldsymbol{v}\rangle|\boldsymbol{u}\rangle\right\|^2 \\ &= \frac{1}{4}\left(2 + \langle \boldsymbol{u}|\boldsymbol{v}\rangle\langle \boldsymbol{v}|\boldsymbol{u}\rangle + \langle \boldsymbol{v}|\boldsymbol{u}\rangle\langle \boldsymbol{u}|\boldsymbol{v}\rangle\right) \\ &= \frac{1 + |\langle \boldsymbol{u}|\boldsymbol{v}\rangle|^2}{2}.\end{aligned} \tag{1.61}$$

This calculation follows the step-by-step presentation in [60] and is also consistent with the standard circuit treatment in Nielsen and Chuang [71, 61]. Repeating the test estimates $|\langle \boldsymbol{u}|\boldsymbol{v}\rangle|^2$ by sampling. The swap test is most useful when the magnitude of an overlap is enough, for example in fidelity estimation.

Remark 1.11 (What the swap test does and does not give)**.** *The swap test gives the squared magnitude of the inner product. It does not give the sign or phase of $\langle \boldsymbol{u}|\boldsymbol{v}\rangle$. If the real or imaginary part is needed and one has access to state-preparation unitaries U_u and U_v, one typically uses a Hadamard-test variant.*

1.9.2 The Hadamard test

Suppose W is a unitary and $|\psi\rangle$ is a state. The Hadamard test estimates $\mathrm{Re}\langle\psi|W|\psi\rangle$ or $\mathrm{Im}\langle\psi|W|\psi\rangle$ by controlling W with one ancilla [60]. It is a concrete instance of the enlarged-register construction in Figure 1.1: the ancilla and system registers undergo the joint controlled unitary

$$|0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes W. \tag{1.62}$$

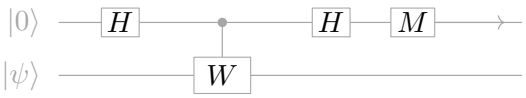

Figure 1.5: Hadamard test for estimating $\mathrm{Re}\,\langle\psi|\,W\,|\psi\rangle$. The box M denotes computational-basis measurement of the ancilla. To estimate the imaginary part, insert $S^\dagger$ before the final Hadamard, where $S = \mathrm{diag}(1, i)$ is the phase gate.

For the real part, the state evolves as

$$\begin{aligned}|0\rangle\,|\psi\rangle &\xmapsto{H\otimes I} \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)\,|\psi\rangle \\ &\xmapsto{\text{controlled-}W} \frac{1}{\sqrt{2}}\left(|0\rangle\,|\psi\rangle + |1\rangle\,W\,|\psi\rangle\right) \\ &\xmapsto{H\otimes I} \frac{1}{2}\,|0\rangle\,(|\psi\rangle + W\,|\psi\rangle) + \frac{1}{2}\,|1\rangle\,(|\psi\rangle - W\,|\psi\rangle).\end{aligned} \tag{1.63}$$

Consequently, after measuring the ancilla, the outcome follows

$$\Pr(0) = \frac{1}{4}\,\||\psi\rangle + W\,|\psi\rangle\|^2 = \frac{1}{2}\left(1 + \mathrm{Re}\,\langle\psi|\,W\,|\psi\rangle\right), \tag{1.64}$$

$$\Pr(1) = \frac{1}{2}\left(1 - \mathrm{Re}\,\langle\psi|\,W\,|\psi\rangle\right), \tag{1.65}$$

and hence

$$\Pr(0) - \Pr(1) = \mathrm{Re}\,\langle\psi|\,W\,|\psi\rangle\,. \tag{1.66}$$

For the imaginary part, apply $S^\dagger$ to the ancilla before the final Hadamard, where

$$S = \begin{bmatrix}1 & 0\\ 0 & i\end{bmatrix}, \qquad S^\dagger = \begin{bmatrix}1 & 0\\ 0 & -i\end{bmatrix} \tag{1.67}$$

is the phase gate. With this convention, the final state becomes

$$\frac{1}{2}\,|0\rangle\,(|\psi\rangle - iW\,|\psi\rangle) + \frac{1}{2}\,|1\rangle\,(|\psi\rangle + iW\,|\psi\rangle), \tag{1.68}$$

so that

$$\Pr(0) = \frac{1}{2}\left(1 + \mathrm{Im}\,\langle\psi|\,W\,|\psi\rangle\right). \tag{1.69}$$

If $U_u\,|0^n\rangle = |\boldsymbol{u}\rangle$ and $U_v\,|0^n\rangle = |\boldsymbol{v}\rangle$, then taking $W = U_u^\dagger U_v$ gives

$$\langle 0^n|\,U_u^\dagger U_v\,|0^n\rangle = \langle \boldsymbol{u}|\boldsymbol{v}\rangle\,. \tag{1.70}$$

Thus the real and imaginary Hadamard tests recover the signed complex inner product, provided controlled versions of the state-preparation circuits are available.

1.10 Postselection and amplitude amplification

Many quantum algorithms produce the desired state only after a favorable measurement outcome. This mechanism is called *postselection*. It is simple but important enough to isolate.

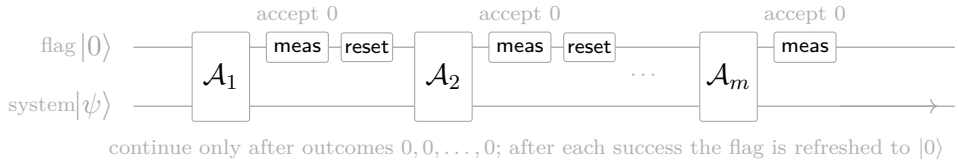

Figure 1.6: Repeated postselection with a reusable flag ancilla. Each box labeled meas is a mid-circuit computational-basis measurement; "accept 0" means that the run continues only when the outcome is 0. After a successful measurement, the flag is refreshed to $|0\rangle$ before the next stage. The total success probability is the product in (1.72).

1.10.1 Postselection

Suppose a circuit prepares

$$\mathcal{A}\,|0\rangle = \sqrt{p}\,|0\rangle\,|\phi\rangle + \sqrt{1-p}\,|1\rangle\,|\phi_\perp\rangle\,, \tag{1.71}$$

where the first register is an ancilla flag. If we measure the flag and keep only the runs in which the outcome is 0, then the remaining register is in the state $|\phi\rangle$. The price is that this happens with probability p. Repeating the experiment until success requires $O(1/p)$ trials on average.

A subtlety that has no exact classical analogue is that postselection can be costly when it is repeated many times. Suppose a computation applies m nonunitary steps and postselects after each one, with success probabilities $p_1, \ldots, p_m$. The probability that the whole run survives is

$$p_{\mathrm{tot}} = \prod_{r=1}^{m} p_r. \tag{1.72}$$

If $p_r \le p < 1$, then $p_{\mathrm{tot}} \le p^m$, which is exponentially small in the number of postselections. Even when the measurements are weak, say $p_r = 1 - \delta_r$, one has the approximation

$$\prod_{r=1}^{m}(1-\delta_r) \approx \exp\left(-\sum_{r=1}^{m}\delta_r\right). \tag{1.73}$$

Thus repeatedly applying a small nonunitary map and postselecting at every step, as pictured in Figure 1.6, can destroy the success probability. This is why many quantum algorithms try to keep the full process coherent.

This operation is analogous to conditioning a random variable on an event, but it is not merely a matter of data analysis after the fact. The unsuccessful quantum runs are discarded, and the state available for subsequent computation is the normalized conditional state. Therefore a postselected subroutine with small success probability can dominate the cost of an algorithm.

Remark 1.12 (Postselection changes normalization)**.** *Before measuring the flag, the desired component in* (1.71) *has norm* $\sqrt{p}$*. After successful postselection, it is renormalized to unit norm. This renormalization is useful because it produces a valid quantum state, but it also hides the magnitude* p*, which must be tracked or estimated separately when it carries physical information.*

1.10.2 Amplitude amplification

A typical situation can be written more abstractly as

$$\mathcal{A}\,|0\rangle = \sqrt{p}\,|\text{good}\rangle\,|\phi\rangle + \sqrt{1-p}\,|\text{bad}\rangle\,|\phi_\perp\rangle\,, \tag{1.74}$$

where p is the success probability. Classical repetition needs $O(1/p)$ attempts. Amplitude amplification boosts the success probability using $O(1/\sqrt{p})$ applications of $\mathcal{A}$ and $\mathcal{A}^\dagger$ [16].

Let $\sin^2\theta = p$, $0 \le \theta \le \pi/2$. Define two reflections: one about the initial state and one that changes the sign of the good subspace. Their product acts as a rotation by angle 2θ in the two-dimensional subspace generated by the good and bad components in (1.74). After k amplification steps, the success probability becomes

$$\sin^2((2k+1)\theta). \tag{1.75}$$

Choosing $k \approx \pi/(4\theta)$ makes the success probability close to one when p is known. When p is unknown, variants based on randomized iteration counts or amplitude estimation are used.

1.10.3 Quantities of interest and recovering the missing norm

Let P_{qoi} be a projector representing a quantity of interest, such as membership in a subdomain or an event in a sampling problem. If

$$p_{\text{qoi}} = \langle\psi|\,P_{\text{qoi}}\,|\psi\rangle\,, \tag{1.76}$$

then estimating p_{qoi} is equivalent to estimating a measurement probability. Standard sampling gives error $O(M^{-1/2})$ after M shots. Amplitude estimation can reduce the error to $O(R^{-1})$ after R coherent uses of the underlying circuit, under stronger circuit assumptions. In PDE applications this matters when p_{qoi} is a physically meaningful scalar output, such as probability mass in a region, a normalized energy fraction, or a success probability associated with a linear-system solver.

The original amplitude-amplification and amplitude-estimation framework of Brassard, Høyer, Mosca, and Tapp gives the basic quadratic improvement from sampling error $O(M^{-1/2})$ after M independent shots to coherent error scaling $O(R^{-1})$ after R circuit uses when a scalar quantity has been encoded as an amplitude [16]. Rall's block-encoding framework for estimating physical quantities gives a useful modern interpretation of the same idea: if an observable, correlation function, density-of-states quantity, or matrix element can be written as a block-encoded matrix element, then a Hadamard test or related overlap circuit turns it into an amplitude, and amplitude estimation gives the quadratic improvement in the target scalar accuracy [75]. This is the viewpoint used later in the PDE chapters when we estimate observables directly instead of first reconstructing a full solution vector.

There is a second issue that is especially important for linear-system and ODE algorithms. These algorithms often prepare a normalized vector

$$|\boldsymbol{v}\rangle = \frac{\boldsymbol{v}}{\|\boldsymbol{v}\|_2}, \tag{1.77}$$

rather than the unnormalized vector $\boldsymbol{v}$ itself. If the desired scalar is a physical amplitude, norm, or energy depending on $\boldsymbol{v}$, then the missing factor $\|\boldsymbol{v}\|_2$ must be recovered separately.

A common situation is that a circuit prepares a flagged state

$$U\left|0^a\right\rangle\left|\boldsymbol{b}\right\rangle = \left|0^a\right\rangle\frac{\boldsymbol{v}}{\alpha} + \left|\Phi_\perp\right\rangle, \qquad (\langle 0^a| \otimes I)\left|\Phi_\perp\right\rangle = 0. \tag{1.78}$$

Measuring the ancilla and observing 0^a succeeds with probability

$$p_{\text{succ}} = \frac{\|\boldsymbol{v}\|_2^2}{\alpha^2}. \tag{1.79}$$

Therefore

$$\|\boldsymbol{v}\|_2 = \alpha\sqrt{p_{\text{succ}}}. \tag{1.80}$$

The success probability can be estimated by sampling or amplitude estimation. Once $\|\boldsymbol{v}\|_2$ is known, normalized measurements can be converted back to unnormalized physical quantities. For example, if O is an observable on the system register, then

$$\boldsymbol{v}^\dagger O\boldsymbol{v} = \|\boldsymbol{v}\|_2^2\,\langle\boldsymbol{v}|\,O\,|\boldsymbol{v}\rangle\,. \tag{1.81}$$

This separation between preparing the direction $\boldsymbol{v}/\|\boldsymbol{v}\|$ and estimating the amplitude $\|\boldsymbol{v}\|$ is often hidden in high-level descriptions of quantum linear solvers. For PDE applications, it should be made explicit because physical quantities such as L^2 norms, energies, fluxes, and residuals are not invariant under normalization.

1.10.4 Amplitude estimation

Postselection and amplitude amplification treat the success probability p in (1.74) as an obstacle. In many PDE applications, however, p *is* the answer: by (1.80), success probabilities encode output norms, and projector expectations $p_\Pi = \langle\psi|\,\Pi\,|\psi\rangle$ encode quantities of interest such as probability mass in a subdomain. Amplitude estimation is the primitive that estimates such a p with quadratically fewer circuit uses than direct sampling [16].

Write $p = \sin^2\theta$ as in amplitude amplification. The product of the two reflections used there is a rotation by 2θ in the two-dimensional good/bad subspace, hence a unitary with eigenvalues $e^{\pm 2i\theta}$. Applying quantum phase estimation, as described in Section 1.8, to this rotation, with the state $\mathcal{A}\left|0\right\rangle$ as input, returns an estimate of θ. Using R controlled applications of $\mathcal{A}$ and $\mathcal{A}^\dagger$, the estimate $\tilde{p} = \sin^2\tilde{\theta}$ satisfies

$$|\tilde{p} - p| \;=\; O\left(\frac{\sqrt{p(1-p)}}{R} + \frac{1}{R^2}\right) \tag{1.82}$$

with constant success probability. Direct sampling with M shots gives error $O(\sqrt{p(1-p)/M})$; matching a target additive error ϵ therefore costs $M = O(\epsilon^{-2})$ samples classically but only $R = O(\epsilon^{-1})$ coherent uses of $\mathcal{A}$. This is the quadratic saving invoked when we say that a scalar output can be estimated to accuracy ϵ with $\widetilde{O}(\epsilon^{-1})$ queries.

Remark 1.13 (What amplitude estimation costs)**.** *The improvement is not free. The R applications of $\mathcal{A}$ occur* coherently *inside one long circuit, so the circuit depth grows by the factor R, and controlled versions of $\mathcal{A}$ are required. When $\mathcal{A}$ itself contains a full PDE solve, this multiplies the solver depth by ϵ^{-1}. Whether amplitude estimation or simple sampling is preferable is therefore a balance between total gate count and circuit depth, the same kind of trade-off as between a single long time integration and an ensemble of short ones.*

1.11 Block encodings

Classical numerical linear algebra is built around matrix-vector multiplication. Quantum algorithms cannot directly apply a nonunitary matrix A to a quantum state, because quantum circuits are unitary. A block encoding solves this by embedding a scaled copy of A into a larger unitary matrix; a detailed scientific-computing treatment is given in [33, 61].

Definition 1.14 (Block encoding)**.** *Let $A \in \mathbb{C}^{N\times N}$. A unitary U acting on $a+n$ qubits is an (α, a, ε) block encoding of A if*

$$\|A - \alpha(\langle 0^a| \otimes I_N)U(|0^a\rangle \otimes I_N)\| \le \varepsilon. \tag{1.83}$$

When $\varepsilon = 0$, we say the block encoding is exact. In block matrix form, exactness means

$$U = \begin{bmatrix} A/\alpha & * \\ * & * \end{bmatrix}, \tag{1.84}$$

with respect to the decomposition in which the ancilla register is $|0^a\rangle$ or orthogonal to $|0^a\rangle$.

The scalar α is called the *subnormalization factor*. A smaller α is better. If U is applied to $|0^a\rangle |\boldsymbol{x}\rangle$, then the component with the ancilla still equal to $|0^a\rangle$ is

$$|0^a\rangle \frac{A\boldsymbol{x}}{\alpha}. \tag{1.85}$$

Measuring the ancilla and postselecting on 0^a succeeds with probability

$$p = \frac{\|A\boldsymbol{x}\|_2^2}{\alpha^2 \|\boldsymbol{x}\|_2^2}. \tag{1.86}$$

Thus the normalization α directly affects the success probability and the complexity of amplitude amplification. In this sense, a block encoding should be viewed as a unitary implementation of a scaled matrix action together with a postselection flag. Without postselection, the full output is still a valid quantum state in the enlarged space; after successful postselection, the system register contains the normalized vector $A\boldsymbol{x}/\|A\boldsymbol{x}\|_2$.

Remark 1.15 (Block encoding as a matrix access model)**.** *A block encoding is not merely a storage format. It is a promise that A/α appears as a submatrix of a unitary that can be implemented efficiently. Thus the real input model is the circuit for U, not the abstract existence of a unitary dilation.*

1.11.1 A small calculus of block encodings

Block encodings can be combined in much the same way that matrices are combined in numerical linear algebra. We record two rules that will be used repeatedly below; for more details, see [33, 61].

First, if U_A is an (α, a, ϵ_A) block encoding of A and U_B is a (β, b, ϵ_B) block encoding of B, then, after placing the ancillas in separate registers, one obtains a block encoding of AB with normalization $\alpha\beta$. More explicitly, on the register order

$$\mathcal{H}_a \otimes \mathcal{H}_b \otimes \mathcal{H}_{\text{sys}},$$

let $\widetilde{U}_A$ denote U_A acting on the a-ancilla register and the system register, with identity on the b-ancilla register, and let $\widetilde{U}_B$ denote U_B acting on the b-ancilla register and the same system register, with identity on the a-ancilla register. In the exact case,

$$(\left\langle 0^{a+b}\right| \otimes I)\widetilde{U}_A\widetilde{U}_B(\left|0^{a+b}\right\rangle \otimes I) = \frac{AB}{\alpha\beta}. \tag{1.87}$$

With errors, the block-encoding error is at most

$$\epsilon_{AB} \leq \alpha\epsilon_B + \beta\epsilon_A + \epsilon_A\epsilon_B. \tag{1.88}$$

This product rule is the quantum analogue of composing two matrix-vector operations, but the normalization factors multiply.

Second, if one has block encodings of $A_0, \ldots, A_{s-1}$, then the (linear combination of unitaries) LCU construction below block encodes a weighted sum

$$A = \sum_{\ell=0}^{s-1} a_\ell A_\ell \tag{1.89}$$

with normalization proportional to $\sum_\ell |a_\ell|\alpha_\ell$, where α_ℓ is the normalization of the ℓth block encoding. Thus the block-encoding calculus looks familiar algebraically, but the normalizations play the role of hidden condition and success-probability parameters.

1.11.2 Linear combination of unitaries

A common way to build block encodings is the linear-combination-of-unitaries (LCU) method. Suppose

$$A = \sum_{\ell=0}^{s-1} a_\ell U_\ell, \qquad a_\ell \geq 0, \tag{1.90}$$

where each U_ℓ is unitary. Let

$$\alpha = \sum_{\ell=0}^{s-1} a_\ell. \tag{1.91}$$

Let $m = \lceil \log_2 s \rceil$, and pad the selector space to dimension 2^m if necessary. Define a preparation unitary P by

$$P\left|0^m\right\rangle = \sum_{\ell=0}^{s-1} \sqrt{a_\ell/\alpha}\left|\ell\right\rangle, \tag{1.92}$$

with zero amplitude on the padded selector labels. Define the unitary

$$\mathrm{SELECT}(U) = \sum_{\ell=0}^{s-1} \left|\ell\right\rangle\left\langle\ell\right| \otimes U_\ell + \sum_{\ell=s}^{2^m-1} \left|\ell\right\rangle\left\langle\ell\right| \otimes I. \tag{1.93}$$

The second sum acts on the padded selector labels and is needed for unitarity; those labels have zero amplitude after PREPARE. In the selector basis, this is simply a block diagonal matrix,

$$\mathrm{SELECT}(U) = \begin{bmatrix} U_0 & & & & \\ & U_1 & & & \\ & & \ddots & & \\ & & & U_{s-1} & \\ & & & & I_{\mathrm{pad}} \end{bmatrix}. \tag{1.94}$$

If $w_\ell = \sqrt{a_\ell/\alpha}$, then the top-left block after preparing and unpreparing the selector register is the weighted compression

$$\begin{bmatrix} w_0 I & w_1 I & \cdots & w_{s-1} I \end{bmatrix} \begin{bmatrix} U_0 & & & \\ & U_1 & & \\ & & \ddots & \\ & & & U_{s-1} \end{bmatrix} \begin{bmatrix} w_0 I \\ w_1 I \\ \vdots \\ w_{s-1} I \end{bmatrix} = \sum_{\ell=0}^{s-1} \frac{a_\ell}{\alpha} U_\ell. \tag{1.95}$$

Thus LCU is not conceptually far from a classical block matrix construction: the selector register chooses one diagonal block, and the preparation vector supplies the coefficients. Then

$$\left(\langle 0^m | P^\dagger \otimes I \right) \mathrm{SELECT}(U) \left(P |0^m\rangle \otimes I \right) = \frac{A}{\alpha}. \tag{1.96}$$

Therefore

$$U_A = (P^\dagger \otimes I)\, \mathrm{SELECT}(U)(P \otimes I) \tag{1.97}$$

block encodes A with subnormalization α.

Negative or complex coefficients are absorbed into the unitaries. For example, $-S$ is unitary if S is unitary.

The coefficient sum

$$\alpha = \sum_{\ell=0}^{s-1} |a_\ell| = \|a\|_1 \tag{1.98}$$

is therefore not merely a normalization convention. Applied to a normalized state $|\psi\rangle$, the LCU block succeeds under postselection with probability

$$p_{\mathrm{LCU}} = \frac{\|A|\psi\rangle\|^2}{\alpha^2}, \tag{1.99}$$

and amplitude amplification introduces a factor of order $\alpha/\|A|\psi\rangle\|$. Thus a decomposition with a large coefficient 1-norm may be expensive even when cancellations make $\|A\|$ small. The quantity α plays a role analogous to a stability or conditioning constant for the chosen decomposition.

Why can LCU nevertheless be useful? Classically, explicitly forming or applying a sum of s unrelated matrices generally requires visiting all s terms. Quantumly, a single SELECT circuit applies the appropriate U_ℓ coherently to every selector branch, and PREPARE supplies all weights in superposition. If PREPARE and SELECT have compact structured circuits, one block-encoding query can therefore represent a sum with many terms while using only $O(\log s)$ selector qubits. This is not an automatic speedup: for unstructured data, preparing the coefficients or implementing the multi-controlled SELECT operation may itself cost $O(s)$. The advantage comes from structure in the coefficient state and in the family $\{U_\ell\}$, as emphasized in [61].

1.11.3 Sparse-matrix block encodings

The phrase "sparse access" can sound mysterious, but it is close in spirit to how sparse matrices are handled in numerical linear algebra. In a classical compressed sparse row format, one stores for each row the list of nonzero column indices and the corresponding values; this is the same

kind of data structure used throughout sparse iterative methods [76]. A quantum sparse-access model asks for coherent versions of these two operations.

Assume $A \in \mathbb{C}^{N\times N}$ has at most s nonzeros in each row and each column. For simplicity suppose the row lists are padded to length s. A row-index oracle returns the column location of the ℓth entry in row i,

$$O_{\mathrm{loc}} : |i\rangle\,|\ell\rangle\,|0\rangle \mapsto |i\rangle\,|\ell\rangle\,|f(i,\ell)\rangle\,, \qquad \ell = 0,\dots,s-1, \tag{1.100}$$

where $f(i,\ell)$ is the column index. A value oracle returns the corresponding matrix entry,

$$O_{\mathrm{val}} : |i\rangle\,|\ell\rangle\,|0\rangle \mapsto |i\rangle\,|\ell\rangle\,\left|A_{i,f(i,\ell)}\right\rangle. \tag{1.101}$$

In one-dimensional finite differences, $f(i,\ell)$ is often just $i-1$, i, or $i+1$. In finite elements, $f(i,\ell)$ is obtained by enumerating the elements touching the basis function φ_i and then the neighboring local basis functions. Thus the oracle is not necessarily a black-box database; it may be a reversible version of the same local indexing logic used in an assembly code.

We now describe one explicit block-encoding construction. Let

$$A_{ij} = |A_{ij}|e^{i\theta_{ij}}, \qquad \alpha \geq \max_i \sum_j |A_{ij}|, \qquad \alpha \geq \max_j \sum_i |A_{ij}|. \tag{1.102}$$

For an s-sparse matrix it is always safe to take $\alpha = s\max_{ij}|A_{ij}|$, consistent with the norm bound (1.22). Introduce an "edge" basis $|i,j\rangle$ for nonzero positions and assume we have two isometries P and Q that prepare the row- and column-weighted square-root states, as follows,

$$P\,|i\rangle = \frac{1}{\sqrt{\alpha}} \sum_{j:A_{ij}\neq 0} \sqrt{|A_{ij}|}\,|i,j\rangle + \sqrt{1-\frac{1}{\alpha}\sum_j |A_{ij}|}\,|i,\perp\rangle\,, \tag{1.103}$$

$$Q\,|j\rangle = \frac{1}{\sqrt{\alpha}} \sum_{i:A_{ij}\neq 0} e^{i\theta_{ij}}\sqrt{|A_{ij}|}\,|i,j\rangle + \sqrt{1-\frac{1}{\alpha}\sum_i |A_{ij}|}\,|\perp,j\rangle\,. \tag{1.104}$$

The garbage states $|i,\perp\rangle$ and $|\perp,j\rangle$ are chosen orthogonal to all edge states and to each other. Then

$$\langle i|\,P^\dagger Q\,|j\rangle = \frac{A_{ij}}{\alpha}, \qquad \text{hence} \qquad P^\dagger Q = \frac{A}{\alpha}. \tag{1.105}$$

If the isometries P and Q are implemented by unitaries that prepare the corresponding edge states from an all-zero work register, then their product gives a block encoding of A/α.

How are the square-root amplitudes in (1.103) and (1.104) implemented? One first creates a uniform superposition over the padded list $\ell = 0,\dots,s-1$, computes $f(i,\ell)$ and $A_{i,f(i,\ell)}$, and then performs a controlled rotation whose sine or cosine is proportional to $\sqrt{|A_{i,f(i,\ell)}|/\alpha}$. The phase $e^{i\theta_{ij}}$ is added by a controlled phase rotation. Uncomputing the value register leaves the desired amplitude in the edge register. This is the sparse analogue of preparing quadrature-weighted local element data in a finite element code.

Remark 1.16 (Relation to standard sparse formats)**.** *Classically, a matrix-vector product with an s-sparse matrix costs $O(sN)$ arithmetic operations because all rows must be visited. Quantumly, the goal is different: construct a unitary whose action on amplitudes has the matrix embedded in one block. The oracle calls in (1.100)–(1.101) are made coherently on superpositions of indices. This is why the cost is counted in oracle queries and reversible arithmetic gates, not in floating-point operations over all rows.*

Remark 1.17 (Structured sparse matrices)**.** *For structured sparse matrices these oracles should not remain abstract. The row locations may be computed by shifts and modular arithmetic, and the values may be constants or simple functions of the grid point. Camps et al. give explicit circuits for block encodings of several structured sparse matrices, including banded circulant and tridiagonal matrices [19]. The finite-difference examples in Section 1.13 are the simplest instances of this principle.*

1.12 Quantum singular value transformation

Once a normalized block A/α is block encoded, QSVT can transform its singular values by a polynomial. In this section, to lighten notation, A denotes the normalized matrix appearing in the block encoding: its singular values lie in $[0, 1]$, and in the Hermitian eigenvalue-transformation setting its eigenvalues lie in $[-1, 1]$. This is the closest quantum analogue of applying a polynomial filter to a matrix in numerical linear algebra; see [33, 61] for the formal construction and a computational-science-oriented exposition.

Let $A \in \mathbb{C}^{N\times N}$ have singular value decomposition

$$A = \sum_k \sigma_k \left|u_k\right\rangle \left\langle v_k\right|, \qquad 0 \le \sigma_k \le 1. \tag{1.106}$$

Suppose U is a block encoding of A. QSVT constructs, from alternating uses of U, $U^\dagger$, and simple phase rotations on an ancilla, a new unitary whose top block is approximately

$$p^{(\mathrm{SV})}(A) := \sum_k p(\sigma_k) \left|u_k\right\rangle \left\langle v_k\right|, \tag{1.107}$$

for a polynomial p satisfying certain parity and boundedness constraints. The foundational theory is due to Gilyén, Su, Low, and Wiebe [33]. For odd polynomials, the transformed block maps right singular vectors to left singular vectors, as displayed. For even polynomials, the natural output is instead $\sum_k p(\sigma_k) \left|v_k\right\rangle \left\langle v_k\right|$, acting within the right singular space. For Hermitian A this distinction disappears in the eigenvalue form (1.109), which is the case used most often in this lecture note.

When A is Hermitian, singular-value transformation can be arranged as an eigenvalue transformation. If

$$A = \sum_k \lambda_k \left|q_k\right\rangle \left\langle q_k\right|, \qquad -1 \le \lambda_k \le 1, \tag{1.108}$$

then the transformed block acts as

$$p(A) = \sum_k p(\lambda_k) \left|q_k\right\rangle \left\langle q_k\right|. \tag{1.109}$$

This Hermitian specialization is often called *quantum eigenvalue transformation*. It is the form most familiar to numerical analysts because it is literally a polynomial functional calculus for a Hermitian matrix.

Remark 1.18 (Analogy with Krylov and Chebyshev methods)**.** *In Krylov methods, one approximates $f(A)\boldsymbol{b}$ by $p_m(A)\boldsymbol{b}$ for a polynomial p_m of degree m [76]. QSVT also implements polynomial approximations, but it does so inside a unitary circuit using a block encoding of A. The degree of p is reflected in the number of calls to the block encoding.*

1.12.1 Example: inverse filters

Let A be Hermitian positive definite and suppose its scaled spectrum lies in

$$\operatorname{spec}(A) \subseteq [1/\kappa, 1]. \tag{1.110}$$

A QSVT-based linear-system algorithm chooses a polynomial p_m such that

$$\max_{x\in[1/\kappa,1]} \left| p_m(x) - \frac{c}{x} \right| \le \varepsilon, \tag{1.111}$$

where one typically takes $c = \Theta(1/\kappa)$. QSVT requires the implemented polynomial to obey the boundedness condition $|p_m(x)| \le 1$ on all of $[-1, 1]$, not only on the spectral interval $[1/\kappa, 1]$. Applying the polynomial to an input state $|\boldsymbol{b}\rangle = \sum_k b_k |q_k\rangle$ produces a successful branch proportional to

$$p_m(A) |\boldsymbol{b}\rangle \approx c \sum_k \frac{b_k}{\lambda_k} |q_k\rangle = cA^{-1} |\boldsymbol{b}\rangle . \tag{1.112}$$

This is the QSVT version of an inverse polynomial filter. The condition number enters because the polynomial must approximate $1/x$ near the endpoint $x = 1/\kappa$ while remaining bounded on $[-1, 1]$. The scaling constant $c = \Theta(1/\kappa)$ also affects the postselection probability: in a worst-case instance the successful branch can have norm smaller by a factor comparable to $1/\kappa$, so output normalization or amplitude amplification can reintroduce condition-number dependence.

Lemma 1.19 (Block encoding of an inverse)**.** *Let $A = A^\dagger$ be positive definite and suppose that U_A is an exact $(\alpha, a, 0)$ block encoding of A, with*

$$\operatorname{spec}(A/\alpha) \subseteq [1/\kappa, 1]. \tag{1.113}$$

For every $0 < \epsilon < 1/2$, QSVT constructs, using

$$O\left(\kappa \log \frac{\kappa}{\epsilon}\right) \tag{1.114}$$

queries to U_A and $U_A^\dagger$, a block encoding whose top-left block B satisfies

$$\left\| B - \frac{\alpha}{\kappa} A^{-1} \right\| \le \epsilon. \tag{1.115}$$

Equivalently, this gives a block encoding of A^{-1} with subnormalization κ/α and scaled error $(\kappa/\alpha)\epsilon$, up to the additional QSVT ancillas. The factor α/κ appears in the successful block because the inverse must be rescaled to remain a contraction. Thus the lemma gives a bounded block encoding of a scaled inverse, not a direct deterministic application of A^{-1}.

Idea. Apply quantum eigenvalue transformation to a polynomial p that approximates $1/(\kappa x)$ on $[1/\kappa, 1]$ while obeying $|p(x)| \le 1$ for all $x \in [-1, 1]$. The required degree is linear in κ up to logarithmic factors [33, 61]. The resulting block acts as $p(A/\alpha) \approx (\alpha/\kappa)A^{-1}$. □

1.12.2 Diagonal matrices and the cost of inversion

A diagonal matrix is the easiest matrix to apply classically. If

$$D = \operatorname{diag}(d_0, d_1, \ldots, d_{N-1}), \tag{1.116}$$

then a classical matrix-vector product merely multiplies each component by d_j. In a quantum block encoding, one must still account for normalization. Suppose that the diagonal entries can be computed reversibly in fixed-point form and that

$$\alpha \geq \max_j |d_j| = \|D\|_2 . \tag{1.117}$$

A block encoding can be constructed by first computing the magnitude and phase $d_j = |d_j| e^{i\phi_j}$, then applying

$$|j\rangle |0\rangle_{\text{val}} |0\rangle_{\text{anc}} \longmapsto |j\rangle |d_j\rangle_{\text{val}} \left(\frac{|d_j|}{\alpha} |0\rangle_{\text{anc}} + \sqrt{1 - \frac{|d_j|^2}{\alpha^2}} |1\rangle_{\text{anc}} \right), \tag{1.118}$$

and finally applying a controlled phase $e^{i\phi_j}$ on the successful branch before uncomputing the value register. The top-left ancilla block is D/α. Thus diagonal matrices are friendly to block encoding when their entries are computable, but the factor α is still part of the algorithmic cost. This magnitude-rotation-plus-phase structure is exactly the diagonal analogue of the sparse construction in Section 1.11.3.

The inverse is more subtle. Assume, after scaling and restricting to the positive-definite real case, that

$$d_j \in [1/\kappa, 1]. \tag{1.119}$$

A generic QSVT inverse filter applied to a block encoding of D must approximate $x \mapsto 1/x$ on $[1/\kappa, 1]$, and therefore has query complexity proportional to the condition number, up to logarithmic factors. In other words, the fact that D is diagonal does not by itself make a block encoding of D^{-1} cheap if the only primitive is a block encoding of D. This point is important for quantum preconditioning. A classical Jacobi preconditioner is almost trivial to apply, but a quantum implementation must still either implement the reciprocal entries coherently or pay the usual inverse-filter cost.

Tong, An, Wiebe, and Lin introduced a different primitive called fast inversion, which directly block encodes inverse eigenvalues by reversible arithmetic when sufficient structure is available [83]. This can be powerful, but it is an additional access assumption. The lesson for numerical analysts is that "diagonal" and "cheap to invert" are not automatically synonymous in the block-encoding model.

1.12.3 Example: heat semigroups and exponential filters

For a parabolic equation after spatial discretization,

$$\frac{du}{dt} = -Lu, \qquad L \succeq 0, \tag{1.120}$$

we need

$$u(t) = e^{-tL}u(0). \tag{1.121}$$

If L/α is block encoded and $\mathrm{spec}(L/\alpha) \subseteq [0,1]$, then one approximates

$$x \mapsto e^{-\alpha t x}, \qquad x \in [0,1]. \tag{1.122}$$

The required polynomial degree grows with the effective time scale αt and with the desired precision. More precisely, the QSVT framework and the associated approximation theory show that $e^{-\beta x}$ on the physical spectral interval $[0,1]$ can be approximated by an admissible QSVT polynomial of degree $O(\sqrt{\beta \log(1/\varepsilon)} + \log(1/\varepsilon))$, so the query count grows like the square root of the effective time scale αt rather than linearly [33, 1]; this is revisited in Chapter 4. The word admissible is important: the implemented polynomial must remain bounded by one on the full signal interval $[-1,1]$, even though it approximates the exponential only on the positive spectral interval. This direct exponential-filter viewpoint is useful conceptually, even though later parabolic chapters also discuss dilation and Gaussian representations with different access-model interpretations.

1.12.4 Example: oscillatory propagators

For a Hermitian Hamiltonian H, Schrödinger evolution is

$$|\psi(t)\rangle = e^{-itH}\,|\psi(0)\rangle\,. \tag{1.123}$$

Hamiltonian simulation can be regarded as implementing polynomial approximations to the real and imaginary parts,

$$\cos(tx), \qquad \sin(tx), \tag{1.124}$$

through the same block-encoding and polynomial-transformation toolkit. More precisely, suppose that H/α has a block encoding. Qubitization, equivalently the Hamiltonian-simulation specialization of QSVT, implements e^{-itH} using

$$O\!\left(\alpha t + \frac{\log(1/\varepsilon)}{\log\log(1/\varepsilon)}\right) \tag{1.125}$$

queries to the block encoding, up to constant-factor conventions in the access model [67, 33]. We usually write this as $\widetilde{O}(\alpha t + \log(1/\varepsilon))$. The leading linear dependence on the scaled time αt is not merely an artifact of the method: in the black-box sparse-Hamiltonian model, no-fast-forwarding lower bounds rule out sublinear dependence for general Hamiltonians [12].

The distinction from the heat example is important. The function $e^{-\beta x}$ is bounded and decaying on $[0,1]$, which allows a square-root dependence on the effective time scale in the polynomial degree. The oscillatory function $e^{-i\tau x}$ is also bounded, but it oscillates on the scale τ; a polynomial must resolve those oscillations, and the generic query complexity is linear in τ. Special structure, such as Fourier diagonalization, may still permit fast-forwarding, but this is an additional property of the operator rather than a generic consequence of block encoding.

For wave equations, this is the route by which a second-order real PDE is converted into a first-order Hermitian dynamics in the hyperbolic chapter.

Remark 1.20 (What QSVT gives)**.** *QSVT gives a new block encoding. If the desired output is a state proportional to $p(A)\,|\boldsymbol{b}\rangle$, one still has to account for postselection, output normalization, and measurement. Thus QSVT solves the spectral transformation problem; it does not by itself solve the data-loading or readout problems.*

1.13 Finite-difference operators as block encodings

We now specialize the preceding constructions to finite-difference matrices. These examples are deliberately elementary. They show how the matrices that appear in standard numerical PDE discretizations can be expressed through a small number of shift unitaries. The stencils below are the standard model finite-difference operators used in introductory numerical PDE texts [53, 84]; Figure 1.7 shows the basic one-dimensional example.

Throughout this section let $N = 2^n$ and define the cyclic shift S by

$$S\,|j\rangle = |j+1 \pmod N\rangle\,, \qquad S^\dagger\,|j\rangle = |j-1 \pmod N\rangle\,. \tag{1.126}$$

The quantum circuit for this shift matrix S is modular addition by one; the circuit for $S^\dagger$ is modular subtraction by one. It should not be confused with the one-qubit phase gate S in (1.67). These are precisely the left- and right-shift components used in explicit block encodings of banded circulant matrices by Camps–Lin–Van Beeumen–Yang [19].

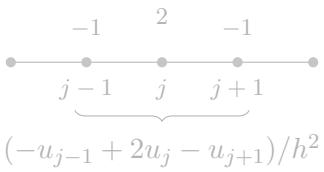

Figure 1.7: The three-point finite-difference stencil for the positive one-dimensional Laplacian $-\partial_{xx}$. The same local stencil is represented quantumly by a short LCU of the identity, the cyclic shift, and the inverse cyclic shift.

1.13.1 Forward difference operator

Consider the periodic forward difference

$$(D_+u)_j = \frac{u_{j+1} - u_j}{h}. \tag{1.127}$$

With our shift convention,

$$D_+ = \frac{S^\dagger - I}{h}. \tag{1.128}$$

Equivalently, $-D_+ = (I - S^\dagger)/h$. Since both I and $-S^\dagger$ are unitary, the LCU construction gives an exact block encoding.

Proposition 1.21 (Block encoding of the forward difference)**.** *Let*

$$\mathrm{SELECT}_D = |0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes (-S^\dagger), \tag{1.129}$$

and let $P_D = H$ on the one-qubit selector register. Then

$$U_D = (H \otimes I)\,\mathrm{SELECT}_D(H \otimes I) \tag{1.130}$$

is an exact $(2/h, 1, 0)$ block encoding of $-D_+$:

$$(\langle 0| \otimes I)U_D(|0\rangle \otimes I) = \frac{I - S^\dagger}{2} = \frac{h(-D_+)}{2}. \tag{1.131}$$

Remark 1.22 (Boundary conditions)**.** *For nonperiodic forward differences, the wrap-around entry must be removed or modified. In the structured sparse approach, this is done by adding boundary-controlled rotations or corrections, exactly as in the tridiagonal nonperiodic modification of the Camps–Lin–Van Beeumen–Yang circulant circuit. Conceptually, one starts from the periodic shift circuit and then zeros out the entries that cross the boundary.*

1.13.2 Three-point Laplacian in one dimension

Let L_h denote the positive periodic one-dimensional discrete Laplacian,

$$(L_h u)_j = \frac{2u_j - u_{j-1} - u_{j+1}}{h^2}. \tag{1.132}$$

Then

$$L_h = \frac{2I - S - S^\dagger}{h^2}. \tag{1.133}$$

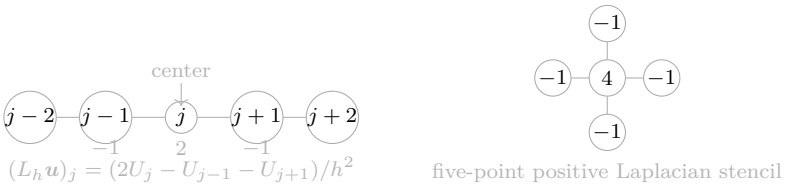

Figure 1.8: The one-dimensional three-point and two-dimensional five-point positive Laplacian stencils. These are the finite-difference matrices later written as LCUs of shift unitaries.

This is the standard three-point stencil for $-\partial_{xx}$ with periodic boundary conditions, shown with its two-dimensional five-point counterpart in Figure 1.8; it is the finite-difference model operator used throughout classical numerical PDE texts [53, 84].

Proposition 1.23 (LCU block encoding of the one-dimensional Laplacian)**.** *Let the selector register have two qubits and define*

$$\mathrm{SELECT}_L = |0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes (-S) + |2\rangle\langle 2| \otimes (-S^\dagger) + |3\rangle\langle 3| \otimes I. \tag{1.134}$$

Let P_L satisfy

$$P_L|0\rangle = \frac{\sqrt{2}}{2}|0\rangle + \frac{1}{2}|1\rangle + \frac{1}{2}|2\rangle. \tag{1.135}$$

The amplitude of $|3\rangle$ is zero. Then

$$U_L = (P_L^\dagger \otimes I)\,\mathrm{SELECT}_L(P_L \otimes I) \tag{1.136}$$

obeys

$$(\langle 0| \otimes I)U_L(|0\rangle \otimes I) = \frac{2I - S - S^\dagger}{4} = \frac{h^2 L_h}{4}. \tag{1.137}$$

Thus U_L is an exact $(4/h^2, 2, 0)$ block encoding of L_h.

Remark 1.24 (Connection with the explicit tridiagonal circuit)**.** *The matrix in* (1.133) *is a banded circulant matrix with stencil coefficients* $a_0 = 2/h^2$ *and* $a_- = a_+ = -1/h^2$ *(the symbols* α*,* γ *are reserved here for the block-encoding normalization and, later, the Schrödinger coefficient). It is therefore a direct specialization of the banded-circulant construction. In the sparse-oracle language,* $f(j, \ell)$ *selects* $j - 1$*,* j*, or* $j + 1$ *modulo* N*, and the value circuit supplies the three stencil weights. The LCU form above is the same structure written in the shorter language of shift unitaries.*

1.13.3 Two-dimensional five-point Laplacian

Let the grid have $N_x = 2^{n_x}$ points in the x direction and $N_y = 2^{n_y}$ points in the y direction. The full state space is

$$\mathbb{C}^{N_x} \otimes \mathbb{C}^{N_y}, \tag{1.138}$$

with basis states $|i\rangle\,|j\rangle$. Define

$$S_x = S \otimes I, \qquad S_y = I \otimes S, \tag{1.139}$$

where each S is the corresponding cyclic shift in one coordinate. For equal mesh spacing h, the positive periodic five-point Laplacian is

$$L_h^{(2D)} = \frac{4I - S_x - S_x^\dagger - S_y - S_y^\dagger}{h^2}. \tag{1.140}$$

Proposition 1.25 (LCU block encoding of the two-dimensional Laplacian)**.** *Let the selector register have three qubits and define SELECT by*

$$\begin{aligned}\mathrm{SELECT}_{2D} = {} & |0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes (-S_x) + |2\rangle\langle 2| \otimes (-S_x^\dagger) \\ & + |3\rangle\langle 3| \otimes (-S_y) + |4\rangle\langle 4| \otimes (-S_y^\dagger) + \sum_{\ell=5}^{7} |\ell\rangle\langle \ell| \otimes I.\end{aligned} \tag{1.141}$$

Let P_{2D} *satisfy*

$$P_{2D}\,|0\rangle = \frac{1}{\sqrt{2}}\,|0\rangle + \frac{1}{\sqrt{8}}\left(|1\rangle + |2\rangle + |3\rangle + |4\rangle\right). \tag{1.142}$$

Then

$$U_{2D} = (P_{2D}^\dagger \otimes I)\,\mathrm{SELECT}_{2D}(P_{2D} \otimes I) \tag{1.143}$$

satisfies

$$(\langle 0| \otimes I)U_{2D}(|0\rangle \otimes I) = \frac{4I - S_x - S_x^\dagger - S_y - S_y^\dagger}{8} = \frac{h^2 L_h^{(2D)}}{8}. \tag{1.144}$$

Thus U_{2D} *is an exact* $(8/h^2, 3, 0)$ *block encoding of* $L_h^{(2D)}$*.*

Remark 1.26 (Anisotropic grid)**.** *For spacings* h_x *and* h_y*,*

$$L_{h_x,h_y}^{(2D)} = \frac{2I - S_x - S_x^\dagger}{h_x^2} + \frac{2I - S_y - S_y^\dagger}{h_y^2}. \tag{1.145}$$

An LCU block encoding has subnormalization

$$\alpha = \frac{4}{h_x^2} + \frac{4}{h_y^2}. \tag{1.146}$$

The selector preparation amplitudes are proportional to the square roots of the absolute stencil coefficients.

1.13.4 General d-dimensional periodic Laplacian

The same construction extends to d dimensions. Let S_r shift the rth coordinate. For a uniform mesh,

$$L_h^{(d)} = \frac{1}{h^2} \sum_{r=1}^{d} (2I - S_r - S_r^\dagger). \tag{1.147}$$

The LCU coefficient sum is

$$\alpha = \frac{4d}{h^2}. \tag{1.148}$$

The block encoding uses a selector register large enough to choose the diagonal term and the $2d$ nearest-neighbor shifts. This is the finite-difference analogue of assembling a stencil matrix from shift matrices.

Remark 1.27 (Cost model)**.** *The cost is dominated by controlled modular additions and subtractions. These operations have circuits of size polynomial in the number of grid qubits. Thus for tensor-product periodic grids, the block encoding has complexity polynomial in* $\log N$*, not polynomial in* N*. Boundary conditions and variable coefficients introduce additional arithmetic and control logic.*

1.14 From finite differences to PDE algorithms

The previous section gave block encodings of discrete differential operators. A PDE algorithm typically adds four more ingredients.

1.14.1 Initial and forcing data

One must prepare amplitude encodings of initial data, forcing terms, or boundary data. Smooth data may be handled by Grover–Rudolph-type preparation. Data produced by a preceding quantum routine may already be in amplitude form. Arbitrary classical data remain expensive unless one assumes a suitable memory model.

1.14.2 Operator functions

Time-dependent PDEs often require functions of matrices. Examples include

$$\text{heat equation:} \qquad u(t) = e^{-tL} u_0, \tag{1.149}$$

$$\text{wave equation:} \qquad u(t) = \cos(t\sqrt{L})u_0 + L^{-1/2}\sin(t\sqrt{L})v_0, \tag{1.150}$$

$$\text{elliptic equation:} \qquad u = L^{-1} f. \tag{1.151}$$

Once L is block encoded and spectrally scaled, QSVT gives a route to approximating these matrix functions by polynomials.

1.14.3 Normalization and success probabilities

If a block encoding of L has subnormalization α, then applying a polynomial $p(L/\alpha)$ produces a normalized quantum output only after postselection or embedded unitary evolution. The norm of the desired vector controls the success probability. This is not a technicality: in PDE problems, smoothing, dissipation, stiffness, and ill-conditioning can strongly affect output norms.

1.14.4 Reading out useful information

A quantum algorithm rarely returns all entries of u. Instead it returns a state proportional to u or allows estimation of selected quantities:

$$\langle g, u\rangle\,, \qquad \|u\|^2_{L^2(\omega)}\,, \qquad \boldsymbol{u}^\dagger B\boldsymbol{u}, \qquad \text{or} \qquad \Pr(x \in \omega). \tag{1.152}$$

This is natural for many PDE tasks but inappropriate if the goal is to print the full solution field.

These four ingredients form the recurring checklist of the lecture note, and the PDE chapters can be read as case studies in how each equation type distributes its cost among them. For elliptic problems, the operator function is an ill-conditioned inverse, and the condition number is not an input parameter but a consequence of the discretization (Chapter 2). For hyperbolic problems, the evolution becomes exactly unitary in the right variables, and the burden shifts to state preparation, sources, and readout (Chapter 3). For parabolic problems, the same smoothing that makes the operator cheap makes the normalized output state expensive (Chapter 4). For nonlinear problems, even the choice of linear representation is a choice of output model (Chapter 5).

1.15 A worked example: Poisson equation on a periodic grid

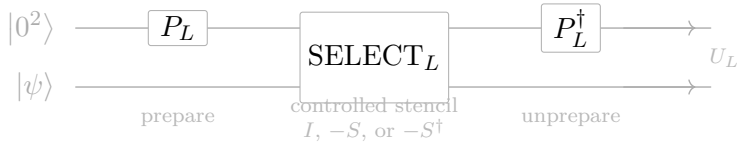

Figure 1.9: Three-stage LCU workflow for the periodic three-point Laplacian. The first box prepares the selector amplitudes, SELECT$_L$ applies I, $-S$, or $-S^\dagger$ conditioned on the selector, and $P_L^\dagger$ unprepares the selector. Projecting the selector onto $|0^2\rangle$ extracts the block $h^2 L_h/4$.

Consider the one-dimensional periodic Poisson problem

$$-u''(x) = f(x), \qquad x \in [0,1], \tag{1.153}$$

with mean-zero right-hand side. The finite-difference discretization is

$$L_h \boldsymbol{u}_h = \boldsymbol{f}_h, \qquad L_h = \frac{2I - S - S^\dagger}{h^2}. \tag{1.154}$$

The constant vector lies in the nullspace, so the inverse is defined on the orthogonal complement of constants.

A quantum linear-system route consists of the following steps.

Step 1. Prepare $|\boldsymbol{f}_h\rangle$, for example using smooth-function state preparation if f is given analytically.

Step 2. Use the block encoding of L_h from Section 1.13; the three-stage circuit is shown in Figure 1.9. Since the block encoding represents L_h/α with $\alpha = 4/h^2$, the nonzero eigenvalues of L_h/α lie in $[\Theta(N^{-2}), 1]$.

Step 3. Use QSVT to approximate $x \mapsto 1/x$ on the nonzero part of the spectrum. The degree depends on the condition number and desired accuracy.

Step 4. Apply the approximate inverse to obtain a state proportional to $\boldsymbol{u}_h = L_h^+ \boldsymbol{f}_h$, where L_h^+ denotes the Moore–Penrose pseudo-inverse on the mean-zero subspace.

Step 5. Estimate quantities of interest using overlap estimation, amplitude estimation, or observable measurements. Convert back to continuum units using the $h^{d/2}$ scaling.

The last step should be read in the sense of Sections 1.4 and 1.10.4. If a normalized solution state is prepared, direct sampling estimates an observable expectation with the usual $O(\epsilon^{-2})$ shot dependence, while amplitude estimation can reduce this to $O(\epsilon^{-1})$ coherent uses of the solution-preparation circuit. The normalization factor hidden by the linear-system postselection must still be recovered if the desired PDE output is an unnormalized quantity such as $\boldsymbol{g}_h^\dagger \boldsymbol{u}_h$ or an L^2 norm.

Remark 1.28 (Conditioning is still present)**.** *Quantum representation does not remove the condition number of the discrete elliptic operator. For the one-dimensional periodic Laplacian, the smallest nonzero eigenvalue is* $\Theta(1)$ *while the largest eigenvalue is* $\Theta(h^{-2})$ *for* L_h*; after scaling by* $\alpha = \Theta(h^{-2})$*, the smallest nonzero scaled eigenvalue is* $\Theta(h^2)$*. Thus inverse approximation still sees a condition number of order* h^{-2}*.*

1.16 A worked example: Schrödinger dynamics with the three-point Laplacian

The Poisson example used the block encoding of L_h to apply an approximate inverse. We now use the same block encoding for a time-dependent problem. Consider the one-dimensional periodic Schrödinger equation with kinetic energy only,

$$i\partial_t u(t,x) = -\gamma \partial_{xx} u(t,x), \qquad x \in [0,1], \tag{1.155}$$

where $\gamma > 0$ is a physical or nondimensional constant. For example, in nondimensional units one may take $\gamma = 1/2$ or $\gamma = 1$. Using the positive periodic discrete Laplacian from (1.133), the semidiscrete equation is

$$i\frac{d}{dt}\boldsymbol{u}_h(t) = H_h \boldsymbol{u}_h(t), \qquad H_h = \gamma L_h = \frac{\gamma}{h^2}\left(2I - S - S^\dagger\right). \tag{1.156}$$

Thus

$$\boldsymbol{u}_h(t) = e^{-itH_h}\boldsymbol{u}_h(0). \tag{1.157}$$

Unlike the Poisson problem, no inverse is involved. The map in (1.157) is unitary because H_h is Hermitian. Therefore the Euclidean norm of the grid vector is preserved:

$$\|\boldsymbol{u}_h(t)\|_2 = \|\boldsymbol{u}_h(0)\|_2 . \tag{1.158}$$

Consequently, if the initial condition is amplitude encoded as

$$|\boldsymbol{u}_h(0)\rangle = \frac{1}{\|\boldsymbol{u}_h(0)\|_2} \sum_{j=0}^{N-1} (\boldsymbol{u}_h(0))_j \, |j\rangle , \tag{1.159}$$

then the desired quantum output is simply

$$|\boldsymbol{u}_h(t)\rangle = e^{-itH_h} \, |\boldsymbol{u}_h(0)\rangle . \tag{1.160}$$

The block encoding of L_h from Section 1.13 satisfies

$$(\langle 0^2| \otimes I) U_L (|0^2\rangle \otimes I) = \frac{h^2 L_h}{4}. \tag{1.161}$$

Hence the same unitary U_L is also a block encoding of the Hamiltonian $H_h = \gamma L_h$, with subnormalization

$$\alpha_H = \frac{4\gamma}{h^2}, \qquad (\langle 0^2| \otimes I) U_L (|0^2\rangle \otimes I) = \frac{H_h}{\alpha_H}. \tag{1.162}$$

It is useful to define the scaled Hermitian matrix

$$A_h := \frac{H_h}{\alpha_H} = \frac{h^2 L_h}{4}. \tag{1.163}$$

The eigenvalues of L_h are

$$\lambda_k(L_h) = \frac{4}{h^2} \sin^2\left(\frac{\pi k}{N}\right), \qquad k = 0, \dots, N-1, \tag{1.164}$$

and therefore

$$\lambda_k(A_h) = \sin^2\left(\frac{\pi k}{N}\right) \in [0,1]. \tag{1.165}$$

Thus the Schrödinger propagator can be written as

$$e^{-itH_h} = e^{-i\tau A_h}, \qquad \tau = \alpha_H t. \tag{1.166}$$

The QSVT viewpoint is now exactly the same as the polynomial functional calculus used in numerical analysis. We want to approximate the scalar function

$$f_\tau(x) = e^{-i\tau x}, \qquad x \in [0,1], \tag{1.167}$$

and then apply the resulting polynomial to $A_h = H_h/\alpha_H$. If p_m is a degree-m polynomial satisfying

$$\max_{x\in[0,1]} \left| p_m(x) - e^{-i\tau x} \right| \le \varepsilon, \tag{1.168}$$

then the corresponding matrix approximation obeys

$$\left\| p_m(A_h) - e^{-i\tau A_h} \right\| \le \varepsilon. \tag{1.169}$$

QSVT, or more specifically its Hamiltonian-simulation specialization often called quantum signal processing or qubitization, compiles such a bounded polynomial transformation into a sequence of phase rotations and alternating calls to the block encoding U_L and its adjoint. Schematically, it constructs a unitary V_Φ such that

$$\left\|(\left\langle 0^{a'}\right| \otimes I)V_\Phi(\left|0^{a'}\right\rangle \otimes I) - p_m(A_h)\right\| \le \varepsilon, \tag{1.170}$$

where a' denotes the selector qubits together with any additional signal or work qubits used by the QSVT construction. Combining (1.168) and (1.170) gives

$$\left\|(\left\langle 0^{a'}\right| \otimes I)V_\Phi(\left|0^{a'}\right\rangle \otimes I) - e^{-itH_h}\right\| \lesssim \varepsilon. \tag{1.171}$$

For numerical analysts, a useful way to view the polynomial approximation underlying Hamiltonian simulation is through the Jacobi–Anger expansion. Although

$$\operatorname{spec}(A_h) \subset [0,1],$$

the signal variable in a QSVT transformation ranges over the full interval $[-1,1]$. It is therefore convenient to approximate the propagator on this full signal interval rather than to approximate only on the physical spectral interval.

For $x \in [-1,1]$, the Jacobi–Anger identities give

$$\cos(\tau x) = J_0(\tau) + 2\sum_{k=1}^{\infty}(-1)^k J_{2k}(\tau)T_{2k}(x), \tag{1.172}$$

$$\sin(\tau x) = 2\sum_{k=0}^{\infty}(-1)^k J_{2k+1}(\tau)T_{2k+1}(x), \tag{1.173}$$

where T_k is the Chebyshev polynomial of the first kind and J_k is the Bessel function of the first kind. Define the truncated polynomials

$$C_R(x) := J_0(\tau) + 2\sum_{k=1}^{R}(-1)^k J_{2k}(\tau)T_{2k}(x), \tag{1.174}$$

$$S_R(x) := 2\sum_{k=0}^{R}(-1)^k J_{2k+1}(\tau)T_{2k+1}(x). \tag{1.175}$$

The polynomial C_R is even and has degree $2R$, whereas S_R is odd and has degree $2R+1$. These are precisely the parity conditions required by the corresponding QSVT transformations.

For every $\delta > 0$, the truncation order can be chosen so that

$$\max_{x\in[-1,1]} |C_R(x) - \cos(\tau x)| \le \delta, \tag{1.176}$$

$$\max_{x\in[-1,1]} |S_R(x) - \sin(\tau x)| \le \delta. \tag{1.177}$$

Consequently,

$$P_R(x) := C_R(x) - iS_R(x) \tag{1.178}$$

approximates $e^{-i\tau x}$ uniformly on $[-1,1]$ with error at most 2δ.

The raw truncations satisfy

$$\|C_R\|_{\infty,[-1,1]}, \|S_R\|_{\infty,[-1,1]} \leq 1 + \delta.$$

A harmless rescaling, for example

$$\widetilde{C}_R = \frac{C_R}{1+\delta}, \qquad \widetilde{S}_R = \frac{S_R}{1+\delta},$$

therefore produces QSVT-admissible real polynomials bounded by one on the full signal interval. QSVT implements the even polynomial $\widetilde{C}_R(A_h)$ and the odd polynomial $\widetilde{S}_R(A_h)$ in separate phase sequences. A one-qubit linear combination of these blocks, followed by robust oblivious amplitude amplification, gives a block encoding of

$$e^{-i\tau A_h} = e^{-itH_h}$$

to the desired precision. This is the block-Hamiltonian simulation construction of Gilyén, Su, Low, and Wiebe [33].

The Chebyshev series above specify the target approximation polynomials; they do not directly give the QSVT phase angles. The latter are obtained from a polynomial-to-phase synthesis procedure.

Remark 1.29 (Parity, boundedness, and complex propagators)**.** *A real QSVT polynomial must have definite parity and must be bounded by one on the full signal interval* $[-1, 1]$*. The Jacobi–Anger truncation* C_R *is even, while* S_R *is odd, so they can be implemented by separate QSVT phase sequences after a small rescaling that enforces the boundedness condition. The complex propagator*

$$e^{-i\tau x} = \cos(\tau x) - i \sin(\tau x)$$

is then obtained by combining the cosine and sine blocks with one additional ancilla and applying robust oblivious amplitude amplification. Thus the full complex exponential is not being treated as one real polynomial of definite parity.

A quantum algorithm for the kinetic Schrödinger equation can therefore be summarized as follows.

Step 1. Prepare the amplitude-encoded initial condition $|\boldsymbol{u}_h(0)\rangle$.

Step 2. Use the LCU block encoding U_L of the three-point Laplacian. For $H_h = \gamma L_h$, the subnormalization is $\alpha_H = 4\gamma/h^2$.

Step 3. Choose a polynomial p_m approximating $x \mapsto e^{-i\alpha_H t x}$ on $[0, 1]$.

Step 4. Use QSVT/QSP to implement the corresponding transformation of the block-encoded matrix H_h/α_H.

Step 5. The resulting system state approximates $|\boldsymbol{u}_h(t)\rangle = e^{-itH_h} |\boldsymbol{u}_h(0)\rangle$.

Step 6. Estimate observables such as probability mass in a subregion, Fourier-mode occupation, overlaps with test functions, or expected position/momentum-type quantities.

Because Schrödinger evolution is unitary, the output norm does not decay or grow. The remaining readout cost is therefore exactly the measurement cost discussed earlier in the chapter: direct sampling repeats the whole simulation for each shot, whereas amplitude estimation uses controlled simulations and their inverses to obtain quadratic improvement in the target scalar accuracy. This distinction will reappear in the wave and heat chapters, where the simulation cost and the measurement cost must be multiplied to obtain the end-to-end complexity.

Remark 1.30 (No fast-forwarding in the general black-box model)**.** *For a generic sparse Hamiltonian specified through black-box access, simulation cannot in general have sublinear dependence on the scaled evolution time: there are sparse Hamiltonians for which constant-accuracy simulation requires $\Omega(\|H\|\, t)$ queries, up to the normalization conventions of the access model [12]. This no-fast-forwarding theorem explains why the generic QSVT/qubitization cost is essentially linear in*

$$\alpha_H t = \frac{4\gamma t}{h^2}. \tag{1.179}$$

Fast-forwarding is therefore a structural exception, not a property of arbitrary sparse matrices.

Remark 1.31 (Why not simply diagonalize by the QFT?)**.** *For this constant-coefficient periodic example, there is also a direct spectral implementation. Since L_h is diagonalized by the discrete Fourier transform,*

$$e^{-itH_h} = F_N^\dagger \operatorname{diag}\left(\exp\left[-it\gamma\frac{4}{h^2}\sin^2\left(\frac{\pi k}{N}\right)\right]\right)_{k=0}^{N-1} F_N. \tag{1.180}$$

A quantum circuit can apply the QFT, compute the eigenvalue and its phase by reversible arithmetic, apply the diagonal phase, and uncompute. Because the matrix is explicitly diagonalizable, this is a structured fast-forwarding mechanism and is not ruled out by the generic lower bound above.

1.17 How to read query and gate counts

Quantum algorithmic complexity is usually reported in several layers. For PDE algorithms, it is helpful to keep the following quantities separate:

- N: number of spatial degrees of freedom, usually with $n = \lceil \log_2 N \rceil$ qubits for the solution register;
- C_{prep}: gate cost of preparing the input state;
- C_U: gate cost of one block-encoding circuit U_A or its inverse;
- Q: number of calls, or queries, to the block encoding;
- α: block-encoding subnormalization factor;
- p_{succ}: postselection success probability;
- M: number of measurement shots used to estimate the desired output quantity.

A typical QSVT-based subroutine has the form

$$|\boldsymbol{b}\rangle \xrightarrow{Q \text{ queries to } U_A} \text{block encoding of } p(A/\alpha) \xrightarrow{\text{postselect}} \frac{p(A/\alpha)\,|\boldsymbol{b}\rangle}{\|p(A/\alpha)\,|\boldsymbol{b}\rangle\|}. \tag{1.181}$$

The leading gate count has the schematic form

$$C_{\text{total}} \approx (C_{\text{prep}} + Q\,C_U + C_{\text{poly}}(Q, n, \log(1/\varepsilon)) + C_{\text{post}})\,C_{\text{meas}}. \tag{1.182}$$

Here C_{poly} denotes the gates needed for phase rotations, arithmetic, and controls; C_{post} accounts for postselection or amplitude amplification; and C_{meas} accounts for repeated sampling or amplitude estimation.

The query count Q is usually comparable to the degree of the polynomial transformation. For example, inverse filters have degree depending on the condition number, while Hamiltonian simulation depends essentially on the scaled time αt. The gate count then multiplies this query count by the cost of implementing the block encoding itself. This is why explicit finite-difference and finite-element block encodings matter: they determine C_U and often also the normalization α.

Remark 1.32 (Comparison with classical complexity)**.** *A classical sparse matrix-vector product is often counted as $O(sN)$ operations. A quantum block encoding of the same sparse matrix may have gate cost polynomial in* $\log N$ *for structured matrices, but it acts on amplitude-encoded vectors and returns limited measurement information. Therefore one should not compare $O(sN)$ directly with* $\text{polylog}(N)$ *without also comparing input preparation, output readout, conditioning, and the quantity of interest.*

Remark 1.33 (Query complexity versus wall-clock runtime)**.** *Query complexity isolates the number of calls to an idealized primitive such as a block encoding or sparse oracle. Gate complexity expands those primitives into elementary gates. Physical runtime further depends on architecture, connectivity, error correction, and parallelism. In this lecture note we mostly discuss query and gate complexity, because these are the architecture-independent quantities most closely connected to numerical linear algebra.*

1.18 Summary

Concept	Numerical linear algebra analogue	Main caution
Quantum state	Unit vector in $\mathbb{C}^N$	Entries are not directly readable
Amplitude encoding	Normalized coefficient vector	Norm is stored separately
Measurement	Sampling an observable's spectral measure	One shot gives an eigenvalue, not an expectation
Postselection	Conditioning on a successful flag	Small success probability is costly
Swap test	Overlap estimator	Gives $\lvert \langle u, v \rangle \rvert^2$, not phase
Hadamard test	Quadratic-form estimator	Needs controlled unitaries
Block encoding	Matrix access model	Subnormalization matters
Diagonal matrix	Entrywise multiplication	Inverse still costs unless reciprocals are encoded
LCU/SELECT	Block diagonal compression	Coefficients enter through normalization
Repeated postselection	Sequential conditioning	Products of probabilities can be exponentially small
Pauli strings	Orthogonal matrix basis	Efficient only when expansion is compact
Matrix norms	Stability and scaling estimates	Max norm and spectral norm differ by dimension
QSVT/QET	Polynomial/eigenvalue matrix function	Degree controls queries
Amplitude amplification	Boosting/postselection	Depends on success probability
Amplitude estimation	Quadratically faster scalar estimation	Long coherent circuits with controlled access
Grover–Rudolph/Kaye–Mosca	Tree-structured data loading	Requires efficient conditional norms/phases
QFT	Coherent Fourier transform	Does not output all Fourier coefficients
Phase estimation	Coherent eigenvalue readout	Aliasing; resolution costs controlled powers
Sparse access	CSR/CSC-like matrix access	Must be implemented reversibly
Query/gate counts	Cost model for primitives	Must include input and measurement

The guiding principle for PDE applications is that discretization, normalization, block encoding, polynomial transformation, postselection, and readout must be analyzed together. A quantum algorithm is not just a faster matrix-vector multiply. It is a full pipeline for preparing data, transforming amplitudes, and extracting limited but useful scalar information.

1.19 Exercises

Exercise 1.1 (Grover–Rudolph tree). *For a four-cell distribution with probabilities p_{00}, p_{01}, p_{10}, p_{11}, write the two levels of conditional rotations explicitly and verify that the final amplitudes are $\sqrt{p_{00}}, \sqrt{p_{01}}, \sqrt{p_{10}}, \sqrt{p_{11}}$.*

Exercise 1.2 (QFT versus FFT). *Let $N = 2^n$. Compare the arithmetic cost of a classical FFT on an explicitly stored vector of length N with the gate count of an exact QFT circuit on n qubits. Explain why this comparison does not mean that the QFT outputs all Fourier coefficients in polylogarithmic time.*

Exercise 1.3 (Phase estimation without aliasing). *Let $H = H^\dagger$ with $\operatorname{spec}(H) \subseteq [0, \lambda_{\max}]$, and set $U = e^{-iHt_0}$. Give a condition on t_0 under which distinct eigenvalues in this window correspond to distinct phases $\theta \in [0, 1)$, and determine how many clock qubits are needed to resolve two eigenvalues separated by $\Delta\lambda$. For the discrete Laplacian with $\lambda_{\max} = \Theta(h^{-2})$, how must t_0 scale with the mesh size?*

Exercise 1.4 (Sparse isometry construction). *Verify (1.105) for a real nonnegative tridiagonal matrix. What changes if the off-diagonal entries are negative?*

Exercise 1.5 (Normalization). *Let $f(x) = \sin(2\pi x)$ on $[0, 1]$ and let $x_j = j/N$. Compute $\|\boldsymbol{f}_h\|_2$ asymptotically and verify the scaling $\|f\|_{L^2} \approx h^{1/2} \|\boldsymbol{f}_h\|_2$.*

Exercise 1.6 (Swap test). *Derive (1.61) directly by expanding the state after the final Hadamard gate.*

Exercise 1.7 (Hadamard-test sign convention). *Repeat the derivation of the imaginary-part Hadamard test. Show that inserting $S^\dagger$ before the final Hadamard gives*

$$\Pr(0) = \frac{1}{2}\left(1 + \operatorname{Im}\langle\psi| W |\psi\rangle\right),$$

whereas inserting S gives the opposite sign. This is a useful reminder that phase conventions in circuit identities should be checked algebraically.

Exercise 1.8 (Observable measurement). *Let $O = \sum_r \lambda_r \Pi_r$ be a Hermitian observable. Starting from the Born probabilities $p_r = \langle\psi| \Pi_r |\psi\rangle$, verify that the expected measurement outcome is $\langle\psi| O |\psi\rangle$.*

Exercise 1.9 (Postselection cost). *For the state in (1.71), compute the expected number of repetitions required to obtain one successful flag measurement. Then compare this with the $O(1/\sqrt{p})$ query scaling from amplitude amplification.*

Exercise 1.10 (Forward difference). *Using $S|j\rangle = |j + 1 \pmod N\rangle$, verify which of $S - I$ or $S^\dagger - I$ corresponds to $(D_+u)_j = (u_{j+1} - u_j)/h$ when vectors are represented by their coefficients in the computational basis.*

Exercise 1.11 (Dirichlet correction). *Write the nonperiodic tridiagonal matrix for the one-dimensional Dirichlet Laplacian. Express it as the periodic circulant Laplacian plus a low-rank boundary correction. Discuss how the correction could be incorporated by LCU.*

Exercise 1.12 (Two-dimensional stencil). *For $N_x = N_y = 4$, explicitly write the five-point periodic Laplacian as a Kronecker sum. Identify the shift matrices S_x and S_y.*

Exercise 1.13 (Condition number). *Compute the eigenvalues of $L_h = (2I - S - S^\dagger)/h^2$ on a periodic grid and estimate the condition number on the mean-zero subspace.*

Exercise 1.14 (Diagonal inverse versus diagonal application). *Let $D = \mathrm{diag}(d_0, \dots, d_{N-1})$ with $d_j \in [1/\kappa, 1]$. Explain why applying D and applying D^{-1} are both $O(N)$ operations classically if all diagonal entries are stored. Then explain why a block encoding of D does not automatically give a constant-query block encoding of D^{-1} in the QSVT model. Relate your answer to the polynomial approximation of $1/x$ on $[1/\kappa, 1]$.*

Exercise 1.15 (Repeated postselection). *Suppose a quantum subroutine consists of m stages and each stage succeeds with probability $1-\delta$. Show that the total success probability is $(1-\delta)^m \approx e^{-m\delta}$ for small δ. If $\delta = c/m$, what constant success probability is obtained in the limit $m \to \infty$? What happens if δ is independent of m?*

Exercise 1.16 (Pauli expansion). *For a one-qubit Hermitian matrix*

$$H = \begin{bmatrix} a & c - id \\ c + id & b \end{bmatrix}, \tag{1.183}$$

with $a, b, c, d \in \mathbb{R}$, write H as a linear combination of I, X, Y, Z. Then verify the coefficient formula (1.14) in this case.

Exercise 1.17 (Recovering an unnormalized norm). *A block-encoding circuit with normalization α prepares the desired vector $\boldsymbol{v}/\alpha$ in the successful ancilla branch. If the success probability is estimated to be p, show that $\|\boldsymbol{v}\| = \alpha\sqrt{p}$. If a normalized-state measurement estimates $\langle \boldsymbol{v}| O |\boldsymbol{v}\rangle$ to accuracy η, how does the uncertainty in p enter the estimate of the unnormalized quadratic form $\boldsymbol{v}^\dagger O \boldsymbol{v}$?*

Exercise 1.18 (Exponentiating a Pauli string). *Let*

$$P = P_{n-1} \otimes \cdots \otimes P_0, \qquad P_j \in \{I, X, Y, Z\}.$$

Show that $P^2 = I$, and hence

$$e^{-i\theta P} = \cos(\theta) I - i \sin(\theta) P.$$

Describe a circuit for this unitary using single-qubit basis changes, a CNOT parity ladder, and one $R_z(2\theta)$ rotation. If P has Pauli weight w, estimate the number of one- and two-qubit gates required.

Exercise 1.19 (Concentration of measurement outcomes). *Let $O = O^\dagger$ have eigenvalues in $[a, b]$, and let $X_1, \dots, X_m$ be independent outcomes obtained by measuring O in the state $|\psi\rangle$. Define*

$$\widehat{\mu}_m = \frac{1}{m} \sum_{r=1}^{m} X_r, \qquad \mu = \langle \psi | O | \psi \rangle.$$

Use the bounded-difference inequality to show that

$$\Pr(|\widehat{\mu}_m - \mu| \geq \epsilon) \leq 2 \exp\left(-\frac{2m\epsilon^2}{(b-a)^2} \right).$$

Specialize this result to a Pauli observable, whose outcomes are ± 1, and determine how many measurements suffice to achieve additive error ϵ with failure probability at most δ.

Exercise 1.20 (Pauli expansion of the one-dimensional Laplacian)**.** *Let $N = 2^n$ and define the noncyclic shift*

$$S_N = \sum_{j=0}^{N-2} |j+1\rangle \langle j| .$$

Using

$$\sigma_+ = |1\rangle \langle 0| = \frac{X - iY}{2}, \qquad \sigma_- = |0\rangle \langle 1| = \frac{X + iY}{2},$$

show that

$$S_N = \sum_{k=0}^{n-1} I^{\otimes(n-k-1)} \otimes \sigma_+ \otimes \sigma_-^{\otimes k}.$$

Expand $S_N + S_N^\dagger$ into Pauli strings and show that it contains at most

$$\sum_{k=0}^{n-1} 2^k = N - 1$$

terms. Conclude that the tridiagonal finite-difference Laplacian

$$L_h = \frac{1}{h^2}\left(2I - S_N - S_N^\dagger\right)$$

has a Pauli expansion with $O(N)$ terms.

Chapter 2

Quantum Algorithms for Elliptic PDEs

2.1 Classical discretization and the direct QLSA route

Elliptic boundary value problems appear whenever a system has reached a static equilibrium. In electrostatics, the potential satisfies a Poisson equation with charge density as the source. In steady heat conduction, the temperature satisfies a diffusion equation with heat sources. In linear elasticity, displacement fields are determined by the balance between elastic stresses and applied loads. These examples have different physical meanings, but they share a common mathematical structure: a coercive second-order operator, a boundary condition, and a variational energy principle. Classical background on elliptic equations may be found in [31] and finite-difference and finite-element discretizations are treated extensively in [53, 69, 17].

The simplest and most familiar model is the Poisson equation. We state the model problem in the variable-coefficient Dirichlet form

$$-\nabla \cdot (a(\boldsymbol{x})\nabla u(\boldsymbol{x})) = f(\boldsymbol{x}), \qquad \boldsymbol{x} \in \Omega, \qquad u|_{\partial\Omega} = 0, \tag{2.1}$$

where $\Omega \subset \mathbb{R}^d$ is bounded and $a(\boldsymbol{x}) \in \mathbb{R}^{d\times d}$ is symmetric and uniformly elliptic: there are constants $0 < a_{\min} \leq a_{\max} < \infty$ such that

$$a_{\min}|\boldsymbol{\xi}|^2 \leq \boldsymbol{\xi}^T a(\boldsymbol{x})\boldsymbol{\xi} \leq a_{\max}|\boldsymbol{\xi}|^2, \qquad \boldsymbol{\xi} \in \mathbb{R}^d, \quad \boldsymbol{x} \in \Omega. \tag{2.2}$$

We may express (2.1) simply as $Lu = f$, where $L = -\nabla \cdot (a\nabla)$ is the elliptic operator. This notation will be used again in the hyperbolic and parabolic chapters: the wave equation evolves according to $u_{tt} + Lu = 0$, while the heat equation evolves according to $u_t + Lu = 0$. Thus the same spatial operator gives rise to three different classes of PDEs.

The Poisson equation is the special case $a(\boldsymbol{x}) = I$:

$$-\Delta u = f, \qquad u|_{\partial\Omega} = 0, \qquad \Delta u = \sum_{j=1}^{d} \frac{\partial^2 u}{\partial x_j^2}. \tag{2.3}$$

The sign convention makes L a positive definite operator under appropriate boundary conditions, e.g., the homogeneous Dirichlet condition. This positivity is the source of stability in the

continuous problem. After discretization, stability is inherited by methods that preserve a discrete analogue of it, in the form of a discrete maximum principle or a coercivity estimate.

Whatever discretization is chosen, the computational problem usually takes the algebraic form

$$A_h \boldsymbol{x}_h = \boldsymbol{b}_h. \tag{2.4}$$

Quantum algorithms targeting (2.4) are known as quantum linear-system algorithms (QLSA). A quantum linear-system algorithm does not normally return every component of $\boldsymbol{x}_h$. Its natural output is instead a normalized state

$$|\boldsymbol{x}_h\rangle := \frac{\boldsymbol{x}_h}{\|\boldsymbol{x}_h\|_2}, \tag{2.5}$$

from which selected overlaps or observables are estimated.

For our discussion of QLSA here, the notation and access model are those of Chapter 1: $\boldsymbol{b}_h$ is amplitude encoded as in (1.33), continuum and Euclidean norms are compared using (1.35), and A_h is supplied through a block encoding in the sense of (1.83). Throughout this chapter, $\boldsymbol{x}_h$ denotes the algebraic vector in the coordinates used for amplitude encoding. In finite differences this is simply the vector of grid values, often denoted $\boldsymbol{u}_h$ locally. In finite elements, we reserve $\boldsymbol{y}$ for the coefficient vector in a nodal basis and use $\boldsymbol{x} = M_h^{1/2}\boldsymbol{y}$ for the mass-normalized coordinate vector. Likewise, $\boldsymbol{b}_h$ denotes the right-hand side after the corresponding coordinate normalization. We use N_h generically for the total number of spatial degrees of freedom when the precise finite-difference or finite-element convention is not important.

Thus one must distinguish the discretization error, the quantum algorithmic error, and the measurement error:

$$u \longrightarrow u_h \longrightarrow |\boldsymbol{x}_h\rangle \longrightarrow \text{quantity of interest}. \tag{2.6}$$

Figure 2.1 previews the most direct route through this pipeline. Its third box announces the theme of the chapter: for elliptic PDEs, the condition number is not an independent input to the linear-algebra problem but a consequence of the discretization, growing as $\Theta(h^{-2})$ exactly when the mesh is refined to meet an accuracy target. This section follows the direct route and pays the full condition number. Section 2.2 exploits the gradient factorization $A_h = G_h^\dagger G_h$ to work at first order, where only the square root of the condition number appears. Section 2.3 bypasses the generic inverse filter altogether when geometry, coefficients, and boundary conditions permit explicit diagonalization; Section 2.4 then treats the associated eigenvalue problem. Across all three routes, one pattern is worth watching for from the start: a saving in the spectral condition number never simply disappears. It moves elsewhere in the pipeline—into an output-block weight, a postselection probability, or the preparation of a transformed input—and an honest end-to-end accounting must follow it there.

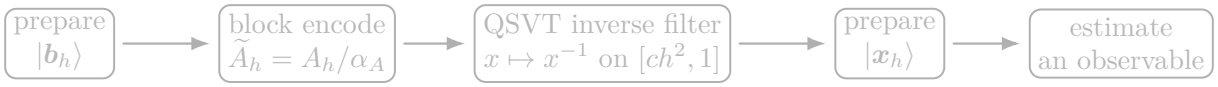

Figure 2.1: The direct elliptic QLSA pipeline. Even if one application of the block encoding costs only polylog(N) gates, the inverse filter must resolve the normalized spectral interval down to $\Theta(h^2)$, producing a query count $\widetilde{O}(h^{-2})$ before state-preparation and readout costs are included.

Notation for sizes. The following table fixes the size notation used in this chapter.

symbol	meaning in this chapter
h	mesh width or grid size
L_{box}	side length of a model tensor-product box
$N_{\mathrm{cell}} = L_{\mathrm{box}}/h$	number of grid cells per coordinate direction
$N_h = N_{\mathrm{cell}} - 1$	number of one-dimensional interior grid unknowns
N_{dof}	total number of spatial degrees of freedom
$N = 2^n$	dimension of a quantum register after power-of-two padding

This small dictionary is included because classical PDE texts often use N for the number of grid points, whereas quantum algorithms often reserve $N = 2^n$ for the dimension of an n-qubit register. Padding to the next power of two is a bookkeeping issue, not a change in the PDE discretization.

2.1.1 Finite differences in one space dimension

Let $\Omega = (0, L_{\mathrm{box}})$ and consider

$$-u''(x) = f(x), \qquad u(0) = u(L_{\mathrm{box}}) = 0. \tag{2.7}$$

Choose a mesh width h with $N_{\mathrm{cell}} = L_{\mathrm{box}}/h \in \mathbb{N}$ and grid points

$$x_j = jh, \quad j = 0, 1, \ldots, N_{\mathrm{cell}}. \tag{2.8}$$

Accordingly, the unknowns are the $N_h = N_{\mathrm{cell}} - 1$ interior values

$$U_j \approx u(x_j), \qquad j = 1, \ldots, N_h.$$

This convention is the one used later in the time-dependent chapters: h is the grid/cell size and L_{box}/h is the number of cells per coordinate direction. For $L_{\mathrm{box}} = 1$, this is equivalent to the familiar notation $h = 1/(N_h + 1)$ if N_h denotes the number of interior unknowns.

The centered second difference gives

$$-\frac{U_{j+1} - 2U_j + U_{j-1}}{h^2} = f(x_j), \qquad j = 1, \ldots, N_h. \tag{2.9}$$

With

$$\boldsymbol{u}_h = (U_1, \ldots, U_{N_h})^\dagger, \qquad (\boldsymbol{b}_h)_j = f(x_j), \tag{2.10}$$

we obtain

$$A_h \boldsymbol{u}_h = \boldsymbol{b}_h, \qquad A_h = \frac{1}{h^2} \begin{bmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{bmatrix}_{N_h \times N_h}. \tag{2.11}$$

This tridiagonal system is the simplest nontrivial elliptic discretization, and it serves as the running example for every quantum construction in this chapter. Three properties of the matrix

A_h organize everything that follows: a stability estimate, an explicit spectrum, and a gradient factorization.

First, Taylor expansion gives a local truncation error $O(h^2)$ for smooth solutions. The discrete maximum principle then converts this consistency estimate into a global maximum-norm error estimate; see, for example, the finite-difference treatments in [53, Chapter 4] and [69]:

$$\max_{1\le j\le N_h} |U_j - u(x_j)| \le Ch^2 \|u^{(4)}\|_{L^\infty(0,L_{\text{box}})}. \tag{2.12}$$

The maximum principle is important conceptually: it is one finite-dimensional analogue of elliptic stability. A quantum algorithm can change how the linear system is solved, but it does not remove the underlying discretization and stability analysis.

Second, the matrix is diagonalized by a discrete sine transform (DST), so its eigenvalues are explicit:

$$\lambda_k(A_h) = \frac{4}{h^2}\sin^2\!\left(\frac{k\pi}{2(N_h+1)}\right), \qquad k = 1,\dots,N_h. \tag{2.13}$$

Consequently, the spectral range is explicit:

$$\lambda_{\min}(A_h) = \Theta(L_{\text{box}}^{-2}), \qquad \lambda_{\max}(A_h) = \Theta(h^{-2}), \qquad \kappa(A_h) = \Theta\!\left(\frac{L_{\text{box}}^2}{h^2}\right). \tag{2.14}$$

For fixed physical domain size $L_{\text{box}} = O(1)$ this reduces to the usual $\Theta(h^{-2})$ condition-number growth. Written in terms of the number of cells per coordinate, $N_{\text{cell}} = L_{\text{box}}/h$, the condition number is $\Theta(N_{\text{cell}}^2)$.

Third, the same matrix is the product of a discrete gradient and a discrete divergence. Let $G_h \in \mathbb{R}^{(N_h+1)\times N_h}$ denote the forward difference with homogeneous boundary values:

$$(G_h \boldsymbol{u}_h)_j = \frac{U_j - U_{j-1}}{h}, \qquad j = 1,\dots,N_h+1, \tag{2.15}$$

where $U_0 = U_{N_h+1} = 0$. Then

$$A_h = G_h^\dagger G_h, \tag{2.16}$$

and

$$\|G_h\| = \Theta(h^{-1}). \tag{2.17}$$

This first-order scale will be exploited in Section 2.2.

2.1.2 The five-point Laplacian on a square

For a square box with the same mesh width in both directions, let N_h denote the number of interior grid points per coordinate. With the unknowns $U_{i,j}$ approximating $u(x_i, x_j)$, the five-point scheme is

$$-\Delta_h U_{i,j} := \frac{4U_{i,j} - U_{i+1,j} - U_{i-1,j} - U_{i,j+1} - U_{i,j-1}}{h^2} = f(x_{i,j}). \tag{2.18}$$

Clearly this is a generalization of the central difference scheme (2.9) to two dimensions.

Equivalently, with appropriate ordering, we can express the matrix via tensor products,

$$A_h = I_{N_h} \otimes T_h + T_h \otimes I_{N_h}, \qquad T_h = \frac{1}{h^2}\,\mathrm{tridiag}(-1, 2, -1). \tag{2.19}$$

The method is again second order for smooth solutions, and its eigenvectors are tensor products of the sine modes from (2.13). The eigenvalues are

$$\lambda_{k,\ell}(A_h) = \frac{4}{h^2}\left[\sin^2\!\left(\frac{k\pi}{2(N_h+1)}\right) + \sin^2\!\left(\frac{\ell\pi}{2(N_h+1)}\right)\right]. \tag{2.20}$$

Thus $\kappa(A_h) = \Theta((L_{\mathrm{box}}/h)^2)$ in any fixed dimension. In d dimensions,

$$A_h = \sum_{m=1}^{d} I_{N_h}^{\otimes(m-1)} \otimes T_h \otimes I_{N_h}^{\otimes(d-m)}. \tag{2.21}$$

Finally, the corresponding discrete gradient can be written explicitly. Let G_1 be the one-dimensional forward-difference matrix from (2.15); then in two dimensions

$$G_h = \begin{bmatrix} I_{N_h} \otimes G_1 \\ G_1 \otimes I_{N_h} \end{bmatrix}, \tag{2.22}$$

up to the convention used for lexicographic ordering. Again one has the scaling,

$$A_h = G_h^\dagger G_h, \qquad \|G_h\| = \Theta(h^{-1}). \tag{2.23}$$

2.1.3 Finite elements, mass normalization, and the stiffness matrix

Finite elements make the variational structure explicit and extend naturally to nonrectangular domains. The weak problem is: find $u \in H_0^1(\Omega)$ such that

$$a(u, v) = \ell(v), \qquad v \in H_0^1(\Omega), \tag{2.24}$$

where

$$a(u, v) = \int_\Omega \nabla v(x)^\dagger a(x) \nabla u(x)\, dx, \qquad \ell(v) = \int_\Omega f(x) v(x)\, dx. \tag{2.25}$$

Let $V_h \subset H_0^1(\Omega)$ have basis $\{\phi_j\}_{j=1}^{N_h}$. Writing

$$u_h(x) = \sum_{j=1}^{N_h} (\boldsymbol{y})_j \phi_j(x), \tag{2.26}$$

the weak form (2.24) implies that

$$K_h \boldsymbol{y} = \boldsymbol{f}_h, \tag{2.27}$$

where

$$(K_h)_{ij} = a(\phi_j, \phi_i), \qquad (\boldsymbol{f}_h)_i = \ell(\phi_i). \tag{2.28}$$

The stiffness matrix K_h is sparse, symmetric, and positive definite after essential boundary conditions are imposed. For conforming P_1 elements on a quasi-uniform mesh, the following estimates are standard in finite element theory; see [53, Chapter 5] and [17]:

$$\|u - u_h\|_{H^1(\Omega)} \le Ch\|u\|_{H^2(\Omega)}, \qquad \|u - u_h\|_{L^2(\Omega)} \le Ch^2\|u\|_{H^2(\Omega)}. \tag{2.29}$$

The coefficient vector $\boldsymbol{y}$ is not an L^2-orthonormal coordinate representation. With the mass matrix

$$(M_h)_{ij} = \int_\Omega \overline{\phi_i(x)}\,\phi_j(x)\,dx, \tag{2.30}$$

we have

$$\|u_h\|^2_{L^2(\Omega)} = \boldsymbol{y}^\dagger M_h \boldsymbol{y}. \tag{2.31}$$

The mass-normalized variables are therefore

$$\boldsymbol{x} = M_h^{1/2}\boldsymbol{y}, \qquad A_h = M_h^{-1/2} K_h M_h^{-1/2}, \qquad \boldsymbol{b}_h = M_h^{-1/2}\boldsymbol{f}_h. \tag{2.32}$$

Then

$$A_h \boldsymbol{x} = \boldsymbol{b}_h, \qquad \|\boldsymbol{x}\|_2 = \|u_h\|_{L^2(\Omega)}. \tag{2.33}$$

These are the coordinates naturally matched to amplitude encoding. They are mathematically convenient, but the exact matrix $M_h^{-1/2}$ is generally not sparse. The situation is nevertheless benign, and the reason is worth recording: on a shape-regular quasi-uniform mesh, the suitably scaled mass matrix is spectrally equivalent to the identity with mesh-independent constants, so $\kappa(M_h) = O(1)$. A QSVT approximation of $x^{-1/2}$ on such a spectrum needs only $O(\log(1/\epsilon))$ degree. Mass normalization is thus a well-conditioned problem hiding inside an ill-conditioned one, and it never dominates the cost.

Similar to the finite difference approach in the previous section, the stiffness matrix also has a Gram factorization. If B_h collects elementwise discrete gradients and W_h is the positive quadrature-weight matrix, then

$$K_h = B_h^\dagger W_h B_h. \tag{2.34}$$

Consequently,

$$A_h = G_h^\dagger G_h, \qquad G_h = W_h^{1/2} B_h M_h^{-1/2}. \tag{2.35}$$

For fixed polynomial degree on quasi-uniform meshes,

$$\|G_h\| = \Theta(h^{-1}), \qquad \|A_h\| = \Theta(h^{-2}), \qquad \kappa(A_h) = \Theta(h^{-2}) \quad \text{on a fixed domain.} \tag{2.36}$$

Thus A_h is not merely a sparse positive matrix: it is the discrete energy operator associated with the weighted gradient G_h.

2.1.4 Direct block encoding of A_h and its mesh-dependent cost

The original quantum linear-system literature focused on optimizing the dependence on the matrix condition number, beginning with HHL and continuing through LCU- and QSVT-based solvers [35, 33]. Elliptic PDEs add an important feature: the condition number is not an independent input parameter. It grows with the number of grid points required by the PDE discretization.

For fixed-order finite differences, and for mass-lumped finite elements in fixed dimension, the normalized operator has $O(1)$ nonzeros per row. The sparse-access construction of Section 1.11.3, or the explicit structured constructions of Section 1.13, then gives a block encoding of

$$\widetilde{A}_h := \frac{A_h}{\alpha_A}, \qquad \alpha_A = \Theta(h^{-2}). \tag{2.37}$$

Because $\lambda_{\min}(A_h) = \Theta(L_{\text{box}}^{-2})$, the normalized spectrum satisfies

$$\operatorname{spec}(\widetilde{A}_h) \subseteq [ch^2/L_{\text{box}}^2, 1] \tag{2.38}$$

for a mesh-independent constant $c > 0$.

The QSVT inverse construction in (1.111) must approximate $1/x$ down to $x = \Theta(h^2/L_{\text{box}}^2)$. Its polynomial degree, and hence the number of calls to the block encoding of A_h, is

$$Q_A = O\left(\kappa(A_h) \log \frac{\kappa(A_h)}{\epsilon}\right) = \widetilde{O}\left(\frac{L_{\text{box}}^2}{h^2} \log \frac{1}{\epsilon}\right). \tag{2.39}$$

This is a query count. The gate count multiplies it by the cost of one block-encoding circuit, as explained in Section 1.17. Meanwhile, state preparation, postselection, and observable estimation remain additional costs.

It is useful to express the same scaling in terms of both the mesh width and the number of unknowns. If $N_{\text{cell}} = L_{\text{box}}/h$ is the number of cells per coordinate and $N_{\text{dof}} = \Theta(N_{\text{cell}}^d)$, then

$$Q_A = \widetilde{O}(N_{\text{cell}}^2) = \widetilde{O}(N_{\text{dof}}^{2/d}) \qquad (L_{\text{box}} \text{ fixed}). \tag{2.40}$$

Thus an efficient, even polylogarithmic, circuit for one sparse-matrix query does not by itself give a polylogarithmic elliptic solver. The inverse filter must still resolve a mesh-dependent spectral interval. If the second-order discretization error is balanced with a target accuracy by taking $h^2 = \Theta(\epsilon)$, then the direct inverse-filter cost is at least $\widetilde{O}(\epsilon^{-1})$ on a fixed box. This mesh dependence is a central difficulty for quantum PDE algorithms and motivates the alternatives in the next two sections.

Finally, quantum registers have dimensions that are powers of two, while a Dirichlet grid usually produces $N_h = N_{\text{cell}} - 1$ unknowns. In practice one pads the matrix and vector to the next power of two, for example by adding an identity block or by adding dummy eigenvalues outside the spectral window of interest. Such padding changes constants but not the h-dependent estimates above.

2.2 First-order factorization and Hermitian dilation

The direct QLSA route treats the second-order stiffness matrix A_h as the primitive operator. Since

$$\kappa(A_h) = \Theta\Big((L_{\text{box}}/h)^2\Big),$$

a direct inverse filter for A_h has the mesh-dependent scale discussed in the previous section. Elliptic structure suggests a more first-order viewpoint. With an appropriate finite difference or finite element discretization, the stiffness matrix has a gradient–divergence factorization,

$$A_h = G_h^\dagger G_h. \tag{2.41}$$

This is the discrete analogue of

$$-\nabla \cdot (a\nabla u) = (\sqrt{a}\,\nabla)^\dagger (\sqrt{a}\,\nabla) u.$$

For instance, the finite-difference factorization in (2.16) and the finite-element factorization in (2.35) are both of this form. The matrix G_h should be thought of as a weighted discrete gradient, while $G_h^\dagger$ is the corresponding negative discrete divergence.

2.2.1 The first-order spectral scale

Recall that L_{box} denotes the characteristic length of the computational domain. For conforming Dirichlet discretizations, the nonzero singular values of G_h satisfy

$$\sigma_{\min}(G_h) = \Theta(L_{\mathrm{box}}^{-1}), \qquad \sigma_{\max}(G_h) = \Theta(h^{-1}), \qquad \kappa_{\mathrm{eff}}(G_h) = \Theta(L_{\mathrm{box}}/h). \tag{2.42}$$

Here κ_{eff} means the ratio of the largest singular value to the smallest nonzero singular value.

It is useful to see this in one dimension. On $(0, L_{\mathrm{box}})$, let G_h be the forward-difference gradient with homogeneous Dirichlet boundary conditions. The sine transform diagonalizes both $G_h^\dagger G_h$ and $G_h G_h^\dagger$, and the singular values are

$$\sigma_k(G_h) = \frac{2}{h} \sin\!\left(\frac{k\pi}{2(N_h+1)}\right), \qquad k = 1, \ldots, N_h. \tag{2.43}$$

Thus

$$\sigma_1(G_h) \sim \frac{\pi}{L_{\mathrm{box}}}, \qquad \sigma_{N_h}(G_h) \sim \frac{2}{h}.$$

This gives (2.42). In several dimensions and on quasi-uniform meshes, the lower bound follows from a discrete Poincaré inequality,

$$\|\boldsymbol{x}\|_2 \le C L_{\mathrm{box}} \|G_h \boldsymbol{x}\|_2,$$

while the upper bound follows from the inverse estimate

$$\|G_h \boldsymbol{x}\|_2 \le C h^{-1} \|\boldsymbol{x}\|_2.$$

Since $A_h = G_h^\dagger G_h$, the first-order relation

$$\kappa_{\mathrm{eff}}(G_h) = \sqrt{\kappa(A_h)} \tag{2.44}$$

is immediate. This square-root relation is the source of the possible improvement, but it is not by itself a complete quantum algorithm. The algorithm must also specify which right-hand side is prepared, which output block is measured, and in which norm the output state is normalized.

2.2.2 Hermitian dilation of the discrete gradient

For a rectangular matrix $G \in \mathbb{C}^{m \times n}$, define its Hermitian dilation by

$$\mathcal{D}(G) = \begin{bmatrix} 0 & G^\dagger \\ G & 0 \end{bmatrix}. \tag{2.45}$$

If

$$G = \sum_j \sigma_j \, |\boldsymbol{u}_j\rangle \, \langle \boldsymbol{v}_j|$$

is a singular-value decomposition, then $\mathcal{D}(G)$ has nonzero eigenvalues $\pm\sigma_j$, with eigenvectors

$$\frac{1}{\sqrt{2}} \begin{bmatrix} \boldsymbol{v}_j \\ \pm \boldsymbol{u}_j \end{bmatrix}.$$

Moreover,

$$\mathcal{D}(G)^{+} = \begin{bmatrix} 0 & G^{+} \\ (G^{\dagger})^{+} & 0 \end{bmatrix}, \tag{2.46}$$

where $^{+}$ denotes the Moore–Penrose pseudo-inverse.

Suppose that G_h/α_G has a block encoding with

$$\alpha_G = \Theta(\|G_h\|) = \Theta(h^{-1}).$$

Then the same access gives a block encoding of $\mathcal{D}(G_h)/\alpha_G$. Its nonzero spectrum lies in

$$\operatorname{spec}\left(\frac{\mathcal{D}(G_h)}{\alpha_G}\right) \setminus \{0\} \subseteq [-1, -c\,h/L_{\text{box}}] \cup [c\,h/L_{\text{box}}, 1]. \tag{2.47}$$

Thus a QSVT pseudo-inverse filter for the dilation has polynomial degree

$$\widetilde{O}(L_{\text{box}}/h),$$

rather than the $\widetilde{O}((L_{\text{box}}/h)^2)$ scale of the direct second-order inverse filter.

The identity

$$\mathcal{D}(G_h)^2 = \begin{bmatrix} G_h^{\dagger}G_h & 0 \\ 0 & G_h G_h^{\dagger} \end{bmatrix}$$

also shows explicitly how the original elliptic operator is embedded in the dilation. However, simply applying $\mathcal{D}(G_h)^{-2}$ would not improve the condition-number dependence. The reason is transparent: squaring the dilation squares its spectrum, so $\kappa(\mathcal{D}(G_h)^2) = \kappa_{\text{eff}}(G_h)^2 = \kappa(A_h)$, and the second-order scale is back. The square-root advantage is available only to algorithms that operate on the dilation itself, at first order. The useful first-order approaches below solve a different, but equivalent, formulation.

2.2.3 Mixed first-order formulation

The most direct PDE formulation is to introduce the discrete flux

$$\boldsymbol{p} = G_h \boldsymbol{x}.$$

Then

$$G_h^{\dagger} G_h \boldsymbol{x} = \boldsymbol{b}$$

is equivalent to

$$\boldsymbol{p} - G_h \boldsymbol{x} = 0, \qquad G_h^{\dagger} \boldsymbol{p} = \boldsymbol{b}. \tag{2.48}$$

In physical terms, $\boldsymbol{p}$ is the flux (in Darcy flow, the velocity): the first equation is the constitutive law and the second is conservation. This pair leads to the Hermitian indefinite system

$$\mathsf{A}_h^{\text{mix}} \begin{bmatrix} \boldsymbol{p} \\ \boldsymbol{x} \end{bmatrix} = \begin{bmatrix} 0 \\ -\boldsymbol{b} \end{bmatrix}, \qquad \mathsf{A}_h^{\text{mix}} = \begin{bmatrix} I & -G_h \\ -G_h^{\dagger} & 0 \end{bmatrix}. \tag{2.49}$$

This system is indefinite, but modern QLSA and QSVT methods can invert Hermitian matrices whose spectra are separated from zero.

There is a small but important scaling issue in (2.49). The identity block is dimensionless, while G_h is a discrete derivative. If L_{box} is varied and the raw matrix is used without scaling, the resulting Euclidean condition number is not a dimensionally invariant quantity. The better formulation is to nondimensionalize the gradient. This observation suggests the following choice of scaling parameter s,

$$s = L_{\text{box}}, \qquad \widetilde{\boldsymbol{p}} = sG_h\boldsymbol{x},$$

and solve

$$\mathsf{A}^{\text{mix}}_{h,s}\begin{bmatrix}\widetilde{\boldsymbol{p}}\\ \boldsymbol{x}\end{bmatrix} = \begin{bmatrix}0\\ -s^2\boldsymbol{b}\end{bmatrix}, \qquad \mathsf{A}^{\text{mix}}_{h,s} = \begin{bmatrix}I & -sG_h\\ -sG_h^\dagger & 0\end{bmatrix}. \tag{2.50}$$

The singular values of sG_h lie in

$$[\,\Theta(1), \Theta(L_{\text{box}}/h)\,].$$

On the two-dimensional subspace associated with a singular value ρ of sG_h, the matrix reduces to

$$\begin{bmatrix}1 & -\rho\\ -\rho & 0\end{bmatrix},$$

whose eigenvalues are

$$\lambda_\pm(\rho) = \frac{1 \pm \sqrt{1+4\rho^2}}{2}. \tag{2.51}$$

Note that $\lambda_+\lambda_- = -\rho^2 < 0$: every singular value contributes one positive and one negative eigenvalue, so the system is genuinely indefinite, with spectrum separated from zero on both sides—exactly the setting that QSVT inverse filters handle. Since $\rho_{\min} = \Theta(1)$ and $\rho_{\max} = \Theta(L_{\text{box}}/h)$, we obtain the meaningful estimate

$$\kappa(\mathsf{A}^{\text{mix}}_{h,L_{\text{box}}}) = \Theta(L_{\text{box}}/h). \tag{2.52}$$

If one ignores this scaling and uses the raw dimensional matrix in (2.49), then the smallest eigenvalue may behave like $\sigma_{\min}(G_h)^2 = \Theta(L_{\text{box}}^{-2})$, while the largest behaves like $\sigma_{\max}(G_h) = \Theta(h^{-1})$. This produces the artifact

$$\kappa(\mathsf{A}^{\text{mix}}_h) = \Theta(L^2_{\text{box}}/h). \tag{2.53}$$

We will regard (2.53) as a warning about units, not as the fundamental PDE scale.

The price of the mixed formulation is that the quantum output lives in an energy-like norm. With the scaled flux variable,

$$\left\|\begin{bmatrix}\widetilde{\boldsymbol{p}}\\ \boldsymbol{x}\end{bmatrix}\right\|_2^2 = \|\boldsymbol{x}\|_2^2 + L^2_{\text{box}}\|G_h\boldsymbol{x}\|_2^2.$$

This is the discrete analogue of

$$\|u\|^2_{L^2(\Omega)} + L^2_{\text{box}}\|\nabla u\|^2_{L^2(\Omega)}.$$

Thus the mixed formulation is natural when fluxes, gradients, stresses, or energy quantities are part of the desired output. If only the potential $\boldsymbol{x}$ is wanted, the probability of projecting onto the $\boldsymbol{x}$-block is

$$p_x = \frac{\|\boldsymbol{x}\|_2^2}{\|\boldsymbol{x}\|_2^2 + L^2_{\text{box}}\|G_h\boldsymbol{x}\|_2^2}. \tag{2.54}$$

For smooth, low-frequency solutions, p_x may be $O(1)$. For mesh-scale components, it can be as small as $O((h/L_{\text{box}})^2)$, so extracting only $\boldsymbol{x}$ can reintroduce the same first-order scale that the mixed system removes from the spectral condition number.

2.2.4 Transformed right-hand sides, Sum-QLS, and normalization

The mixed formulation solves a coupled first-order PDE system. The factorized QLS framework of Orsucci and Dunjko gives a different access-model route: instead of solving the mixed saddle-point system, it applies the pseudo-inverse of the rectangular factor G_h to a suitably transformed right-hand side [72]. Their work also explains why positive definiteness alone is not enough to guarantee a $\sqrt{\kappa}$-type quantum speedup in the standard black-box QLSA model.

In the notation of this chapter, choose a generalized right inverse

$$R_h \in \mathbb{C}^{m_h \times N_h}$$

of $G_h^\dagger$, satisfying

$$G_h^\dagger R_h = I, \qquad \boldsymbol{b}'_h := R_h \boldsymbol{b}_h. \tag{2.55}$$

Then the original solution is recovered from the least-squares problem

$$\boldsymbol{x}_* = \operatorname*{argmin}_{\boldsymbol{x}} \|G_h \boldsymbol{x} - \boldsymbol{b}'_h\|_2. \tag{2.56}$$

Indeed,

$$\begin{aligned} \boldsymbol{x}_* &= G_h^+ \boldsymbol{b}'_h = G_h^+ R_h \boldsymbol{b}_h \\ &= (G_h^\dagger G_h)^{-1} G_h^\dagger R_h \boldsymbol{b}_h = A_h^{-1} \boldsymbol{b}_h. \end{aligned} \tag{2.57}$$

Equivalently, using the Hermitian dilation,

$$\mathcal{D}(G_h)^+ \begin{bmatrix} 0 \\ \boldsymbol{b}'_h \end{bmatrix} = \begin{bmatrix} G_h^+ \boldsymbol{b}'_h \\ 0 \end{bmatrix} = \begin{bmatrix} \boldsymbol{x}_* \\ 0 \end{bmatrix}. \tag{2.58}$$

Because G_h is rectangular, only the component of $\boldsymbol{b}'_h$ in $\mathrm{Range}(G_h)$ contributes. Let

$$\Pi_{G_h} := G_h G_h^+$$

and, for the normalized state $|\boldsymbol{b}'_h\rangle$, define

$$\gamma_h := \|\Pi_{G_h} |\boldsymbol{b}'_h\rangle\|^2. \tag{2.59}$$

The component orthogonal to $\mathrm{Range}(G_h)$ is annihilated by G_h^+. Hence a pseudo-inversion algorithm must project onto or amplify the useful component, which introduces an overhead proportional to $1/\sqrt{\gamma_h}$.

The crucial additional cost is the preparation of $\boldsymbol{b}'_h = R_h \boldsymbol{b}_h$. Suppose, for example, that R_h/α_R has a block encoding and that $|\boldsymbol{b}_h\rangle$ can be prepared. Applying the block encoding of R_h/α_R to $|\boldsymbol{b}_h\rangle$ produces the successful branch

$$\frac{R_h |\boldsymbol{b}_h\rangle}{\alpha_R},$$

with probability

$$p_R = \frac{\|R_h \boldsymbol{b}_h\|_2^2}{\alpha_R^2 \|\boldsymbol{b}_h\|_2^2}. \tag{2.60}$$

Thus preparing the normalized transformed state costs, schematically,

$$C_{Rb} = \widetilde{O}\left((C_b + C_R)\frac{\alpha_R\|\boldsymbol{b}_h\|_2}{\|R_h\boldsymbol{b}_h\|_2}\right), \tag{2.61}$$

if amplitude amplification is used. Here C_b is the cost of preparing $|\boldsymbol{b}_h\rangle$ and C_R is the cost of applying the block encoding of R_h/α_R. If R_h itself is implemented by a QSVT inverse filter for a global elliptic operator, then this cost may already contain the condition number of the original problem and the factorized advantage is lost. Thus R_h must be cheap because of additional structure, not merely because it exists algebraically.

This point is visible in the Moore–Penrose choice. The Moore–Penrose right inverse of $G_h^\dagger$ is

$$R_h = (G_h^\dagger)^+ = G_h(G_h^\dagger G_h)^{-1} = G_h A_h^{-1}. \tag{2.62}$$

With this choice,

$$R_h\boldsymbol{b}_h = G_h\boldsymbol{x}_*, \tag{2.63}$$

which is precisely the unknown flux. It has perfect range overlap $\gamma_h = 1$, but preparing it directly would require solving the elliptic problem. The two extremes bracket the design space: the Moore–Penrose choice is perfectly aligned with Range(G_h) but as hard as the original problem, while a generic cheap R_h may waste most of its amplitude outside the range. A useful R_h must sit in between, structured enough to be applied cheaply and aligned enough that γ_h is not small.

The Sum-QLS construction of Orsucci and Dunjko identifies settings where a nontrivial R_h is cheap. In their model, the matrix is supplied as a sum of local strictly positive-definite terms,

$$A = \sum_{j=1}^{J} H^{(j)}, \qquad H^{(j)} = L^{(j)}L^{(j)\dagger}. \tag{2.64}$$

They form

$$L = \begin{bmatrix} L^{(1)} & \cdots & L^{(J)} \end{bmatrix}, \qquad R = \frac{1}{J}\begin{bmatrix} (L^{(1)})^{-1} \\ \vdots \\ (L^{(J)})^{-1} \end{bmatrix}, \tag{2.65}$$

so that $LL^\dagger = A$ and $LR = I$. Under their access assumptions, the local inverses $(L^{(j)})^{-1}$ are small or structured enough that $R\boldsymbol{b}$ can be prepared efficiently. In that situation, the pseudo-inverse step has the first-order scale

$$\widetilde{O}\left(\frac{\kappa_{\text{eff}}(G_h)}{\sqrt{\gamma_h}}\right) = \widetilde{O}\left(\frac{L_{\text{box}}/h}{\sqrt{\gamma_h}}\right), \tag{2.66}$$

in addition to the transformed-state preparation cost.

Finite element assembly has a superficially similar local decomposition into element contributions. The analogy is useful, but the hypotheses are different: standard element stiffness matrices are typically positive semidefinite and rank deficient, and the number of elements grows with the number of degrees of freedom. Therefore the specific Sum-QLS theorem does not apply automatically to a general finite element matrix. The mixed formulation is often the more direct PDE construction; the Sum-QLS viewpoint becomes attractive when the generalized right inverse can be applied cheaply.

2.2.5 Comparison with classical elliptic solvers

Recall from the size table in section 2.1 that

$$N_{\text{cell}} = \frac{L_{\text{box}}}{h}, \qquad N_{\text{dof}} \asymp N_{\text{cell}}^d. \tag{2.67}$$

Then

$$\kappa(A_h) = \Theta(N_{\text{cell}}^2), \qquad \kappa_{\text{eff}}(G_h) = \Theta(N_{\text{cell}}). \tag{2.68}$$

The following table gives a schematic, fixed-access comparison. For quantum algorithms, the middle columns show query scales to normalized block encodings, before gate costs, input preparation, postselection, and measurement. For classical algorithms, they show arithmetic work. The output factors χ_A and χ_{fac} appearing in the last column are defined at the end of the subsection.

For a dimension-aware comparison, let

$$N_h := \frac{L_{\text{box}}}{h} - 1, \qquad N_{\text{dof}} \asymp N_h^d, \tag{2.69}$$

where N_h is the number of mesh cells, or mesh intervals, per coordinate direction, while N_{dof} is the total number of spatial degrees of freedom. Thus

$$\kappa(A_h) = \Theta(N_h^2), \qquad \kappa_{\text{eff}}(G_h) = \Theta(N_h).$$

The table below gives a schematic comparison. For the quantum rows, the scale is the number of queries to a normalized block encoding, before gate costs, state preparation, postselection, and measurement are included. For the classical rows, the scale is arithmetic work for producing the full discrete vector.

method	scale in N_h	scale in N_{dof}	additional factor to track
direct QSVT on A_h	$\widetilde{O}(N_h^2)$	$\widetilde{O}(N_{\text{dof}}^{2/d})$	normalization χ_A
mixed first-order system	$\widetilde{O}(N_h)$	$\widetilde{O}(N_{\text{dof}}^{1/d})$	energy norm and x-block probability
classical CG, no preconditioner	$\widetilde{O}(N_h^{d+1})$	$\widetilde{O}(N_{\text{dof}}^{1+1/d})$	sparse matrix-vector products
classical multigrid/BPX/PCG	$\widetilde{O}(N_h^d)$	$\widetilde{O}(N_{\text{dof}})$	problem-dependent hierarchy

The direct QSVT row has the usual output-normalization factor

$$\chi_A := \frac{\|\boldsymbol{b}_h\|_2}{\|A_h^{-1}\boldsymbol{b}_h\|_2}. \tag{2.70}$$

For sufficiently accurate L^2-normalized discretizations, this ratio approaches the continuum quantity

$$\frac{\|f\|_{L^2(\Omega)}}{\|u\|_{L^2(\Omega)}}.$$

For the mixed formulation, the natural output is not just $\boldsymbol{x}_*$, but the scaled energy variable

$$\begin{bmatrix} \widetilde{\boldsymbol{p}}_* \\ \boldsymbol{x}_* \end{bmatrix}, \qquad \widetilde{\boldsymbol{p}}_* = L_{\text{box}} G_h \boldsymbol{x}_*.$$

Its norm is

$$\left\| \begin{bmatrix} \widetilde{\boldsymbol{p}}_* \\ \boldsymbol{x}_* \end{bmatrix} \right\|_2^2 = \|\boldsymbol{x}_*\|_2^2 + L_{\mathrm{box}}^2 \|G_h \boldsymbol{x}_*\|_2^2. \tag{2.71}$$

This is the discrete analogue of

$$\|u\|_{L^2(\Omega)}^2 + L_{\mathrm{box}}^2 \|\nabla u\|_{L^2(\Omega)}^2.$$

Thus the mixed system is naturally normalized in an H^1-type energy norm. If the desired output is only the potential block $\boldsymbol{x}_*$, the probability of obtaining that block is given by equation (2.54).

The table should not be read as a claim of quantum advantage. The quantum entries are idealized block-encoding query counts, while the classical entries are full-vector arithmetic costs. Classical multigrid, domain decomposition, and BPX-type preconditioners already achieve nearly linear complexity for many standard elliptic problems [76, 15]. The meaningful quantum question is more specific: after input preparation, block-encoding construction, output normalization, and measurement are included, can a quantum method improve the scaling for a particular quantity of interest?

2.3 QFT-based spectral filtering for special elliptic problems

Fourier methods are one of the oldest tools in the analysis of Poisson's equation: separation of variables turns the PDE into scalar algebraic equations for Fourier coefficients. The quantum version preserves this same idea, but performs the change of basis coherently on amplitudes. This gives a third route, different in kind from the previous two. It applies only when the geometry, coefficients, and boundary conditions make the elliptic operator explicitly diagonalizable. In that setting, one need not query a generic block encoding of A_h and approximate its inverse by a QSVT polynomial. Instead, one transforms to a known eigenbasis and evaluates the reciprocal eigenvalues by reversible arithmetic.

2.3.1 Fourier diagonalization

Consider first the constant-coefficient periodic Poisson equation on the flat torus $\mathbb{T}^d$ that came from periodic boundary conditions,

$$-\Delta u = f, \qquad \int_{\mathbb{T}^d} f(x)\, dx = 0. \tag{2.72}$$

The zero-mean condition removes the nullspace with constant functions. Using the Fourier convention $e^{2\pi i \boldsymbol{k}\cdot x}$, one obtains

$$\widehat{u}_{\boldsymbol{k}} = g(\boldsymbol{k})\widehat{f}_{\boldsymbol{k}}, \qquad g(\boldsymbol{k}) = \frac{1}{4\pi^2|\boldsymbol{k}|^2}, \qquad \boldsymbol{k} \neq \boldsymbol{0}. \tag{2.73}$$

If the convention $e^{i\boldsymbol{k}\cdot x}$ is used instead, the factor $4\pi^2$ is absent. We write it explicitly here to avoid ambiguity.

For a periodic finite-difference grid with $N = 2^n$ points per coordinate, the QFT diagonalizes the circulant Laplacian:

$$A_h = F_h^\dagger \Lambda_h F_h, \tag{2.74}$$

where, in one dimension, we have the familiar eigenvalues,

$$(\Lambda_h)_{kk} = \frac{4}{h^2}\sin^2\!\left(\frac{\pi k}{N}\right). \tag{2.75}$$

The multidimensional symbol is the sum over coordinate directions. Homogeneous Dirichlet and Neumann problems on rectangular boxes lead similarly to sine and cosine transforms. Thus the QFT route is best understood as a structured-grid spectral method, not as a general elliptic solver.

Because the periodic Laplacian has a zero mode, the inverse below is always the pseudo-inverse on the mean-zero subspace. For data satisfying the compatibility condition in (2.72), the constant Fourier mode is absent and this distinction is harmless. A QFT-based solver consists of three operations:

$$|\boldsymbol{b}_h\rangle \xrightarrow{F_h} \left|\widehat{\boldsymbol{b}}_h\right\rangle \xrightarrow{\Lambda_h^+} \Lambda_h^+ \left|\widehat{\boldsymbol{b}}_h\right\rangle \xrightarrow{F_h^\dagger} A_h^{-1}\,|\boldsymbol{b}_h\rangle\,. \tag{2.76}$$

Here Λ_h^+ denotes the pseudo-inverse: the zero Fourier mode is mapped to zero. Because the right-hand side in (2.72) has zero mean, that mode has zero amplitude and the pseudo-inverse agrees with the physical solution operator on the relevant subspace. The QFT on each n-qubit coordinate register uses $O(n^2)$ elementary gates, or $O(dn^2)$ gates in d dimensions. The nonunitary reciprocal filter can be implemented as a diagonal block encoding, using the diagonal-matrix construction discussed in Subsection 1.12.2.

2.3.2 Direct arithmetic block encoding of the reciprocal filter

The middle arrow in (2.76) is a diagonal-matrix block encoding of the kind introduced in Subsection 1.12.2. We spell it out here because this is the only nontrivial step and where the condition number reemerges.

Let $\lambda_{\min}$ and $\lambda_{\max}$ denote the smallest and largest nonzero diagonal entries of Λ_h, so that

$$\kappa = \frac{\lambda_{\max}}{\lambda_{\min}} = \Theta(h^{-2})$$

on a fixed domain. The zero Fourier mode is assigned reciprocal value zero, which is the pseudo-inverse convention for the periodic Poisson problem. Define

$$r_{\boldsymbol{k}} = \begin{cases} 1/(\Lambda_h)_{\boldsymbol{k}\boldsymbol{k}}, & \boldsymbol{k} \neq \boldsymbol{0}, \\ 0, & \boldsymbol{k} = \boldsymbol{0}, \end{cases} \qquad \rho_{\boldsymbol{k}} := \frac{r_{\boldsymbol{k}}}{\alpha_{\mathrm{inv}}}, \qquad \alpha_{\mathrm{inv}} \geq \|\Lambda_h^+\| = \lambda_{\min}^{-1}. \tag{2.77}$$

Thus $0 \leq \rho_{\boldsymbol{k}} \leq 1$. The scale $\alpha_{\mathrm{inv}} = \Theta(\lambda_{\min}^{-1})$ is the diagonal analogue of the normalization factor in a block encoding of A_h^{-1}. It is also the place where the condition number enters the postselection probability.

The arithmetic implementation uses a reversible eigenvalue-inversion oracle. More precisely, let U_ρ be a unitary acting on the Fourier-mode register and a work register such that, up to fixed-point arithmetic error,

$$U_\rho : |\boldsymbol{k}\rangle\,|0\rangle_{\mathrm{val}} \longmapsto |\boldsymbol{k}\rangle\,|\widetilde{\rho}_{\boldsymbol{k}}\rangle_{\mathrm{val}}\,.$$

The first step is therefore

$$|\boldsymbol{k}\rangle\,|0\rangle_{\mathrm{val}}\,|0\rangle_a \longmapsto |\boldsymbol{k}\rangle\,|\widetilde{\rho}_{\boldsymbol{k}}\rangle_{\mathrm{val}}\,|0\rangle_a\,. \tag{2.78}$$

This is the structured-grid version of the "fast inversion" primitive: the eigenvalue is not learned by phase estimation, but computed directly from the explicit Fourier formula for $(\Lambda_h)_{\boldsymbol{k}\boldsymbol{k}}$. This is why the construction is special to diagonalizable tensor-product problems.

Next, a controlled rotation acts on the ancilla qubit. If

$$R_y(\theta)\,|0\rangle = \cos(\theta/2)\,|0\rangle + \sin(\theta/2)\,|1\rangle\,,$$

then choosing

$$\theta_{\boldsymbol{k}} = 2\arccos(\widetilde{\rho}_{\boldsymbol{k}})$$

gives

$$|\widetilde{\rho}_{\boldsymbol{k}}\rangle_{\mathrm{val}}\,|0\rangle_a \longmapsto |\widetilde{\rho}_{\boldsymbol{k}}\rangle_{\mathrm{val}}\left(\widetilde{\rho}_{\boldsymbol{k}}\,|0\rangle_a + \sqrt{1-\widetilde{\rho}_{\boldsymbol{k}}^2}\,|1\rangle_a\right). \tag{2.79}$$

This is the same reciprocal-controlled-rotation idea that appears in HHL and in the post-QPE view of linear-system algorithms: the reciprocal is encoded as the amplitude of a successful ancilla branch. The difference here is that the reciprocal is obtained by reversible arithmetic from the Fourier index $\boldsymbol{k}$, rather than from a phase-estimation register.

Now apply $U_\rho^\dagger$ to uncompute the value register. On a Fourier-space input

$$\left|\widehat{\boldsymbol{b}}_h\right\rangle = \sum_{\boldsymbol{k}} \widehat{b}_{\boldsymbol{k}}\,|\boldsymbol{k}\rangle\,,$$

the state before the final ancilla measurement has the form

$$\sum_{\boldsymbol{k}} \widehat{b}_{\boldsymbol{k}}\,|\boldsymbol{k}\rangle\,|0\rangle_{\mathrm{val}}\left(\rho_{\boldsymbol{k}}\,|0\rangle_a + \sqrt{1-\rho_{\boldsymbol{k}}^2}\,|1\rangle_a\right),$$

where arithmetic errors have been suppressed. Therefore, conditioned on measuring the ancilla in $|0\rangle_a$, the successful branch is proportional to

$$\sum_{\boldsymbol{k}} \rho_{\boldsymbol{k}}\widehat{b}_{\boldsymbol{k}}\,|\boldsymbol{k}\rangle = \frac{\Lambda_h^+}{\alpha_{\mathrm{inv}}}\left|\widehat{\boldsymbol{b}}_h\right\rangle. \tag{2.80}$$

After applying the inverse QFT, this gives a block-encoding implementation of $A_h^+/\alpha_{\mathrm{inv}}$ on the mean-zero subspace.

The success probability is

$$p_{\mathrm{spec}} = \sum_{\boldsymbol{k}} |\rho_{\boldsymbol{k}}|^2|\widehat{b}_{\boldsymbol{k}}|^2 = \frac{\|A_h^+\boldsymbol{b}_h\|_2^2}{\alpha_{\mathrm{inv}}^2\|\boldsymbol{b}_h\|_2^2}.$$

This formula is the useful warning. Even though the diagonal reciprocal is computed directly by arithmetic, the normalization $\alpha_{\mathrm{inv}} \simeq \lambda_{\min}^{-1}$ can make the successful branch small. For a right-hand side concentrated in high-frequency modes, the amplitude can be as small as $O(1/\kappa)$, so amplitude amplification can reintroduce a condition-number cost. For smooth or low-frequency data, the Fourier coefficients $\widehat{b}_{\boldsymbol{k}}$ are concentrated where $r_{\boldsymbol{k}}$ is large, and the effective success probability can be much better. This is the spectral-filtering advantage exploited by the structured QFT approach.

This arithmetic construction is closely related to the diagonal inverse block encodings discussed in Chapter 1 and to the fast-inversion primitive of Tong, An, Wiebe, and Lin [83]. It bypasses a

generic QSVT polynomial approximation to $1/x$ by using an explicit formula for the eigenvalues. The tradeoff is that it depends on separability, efficient reversible evaluation of the diagonal entries, and the same normalization and postselection issues that appear in other block-encoding constructions.

Finally, the factorization viewpoint is also visible in Fourier space. For the periodic Laplacian,

$$(\Lambda_h)_{\boldsymbol{k}\boldsymbol{k}} = \sum_{r=1}^{d} |g_r(\boldsymbol{k})|^2,$$

where $g_r(\boldsymbol{k})$ is the Fourier symbol of the discrete gradient in the r-th coordinate. Thus the eigenvalues are squares of first-order symbols. One could therefore implement first-order filters in terms of $\sqrt{(\Lambda_h)_{\boldsymbol{k}\boldsymbol{k}}}$, paralleling the G_h-based factorization in Section 2.2. For solving Poisson directly, however, the reciprocal $1/\lambda_{\boldsymbol{k}}$ must still be applied or encoded through a suitable mixed or transformed formulation. The QFT arithmetic method makes this distinction transparent: it can compute either $\lambda_{\boldsymbol{k}}^{-1}$ or $\lambda_{\boldsymbol{k}}^{-1/2}$, but the final normalization and the desired output determine which construction is useful.

The framework of Huang, Antonioli, and Barbaresco develops this construction systematically for constant-coefficient elliptic, Helmholtz, and diffusion problems [36]. Its principal quantum contribution is not the Fourier diagonalization—that part is classical—but a modular reversible-arithmetic circuit for block encoding frequency-dependent filters and reciprocals, building on fast inversion for explicitly computable diagonal entries [83]. Under efficient arithmetic oracles, the schematic coherent-filter cost is of the form

$$C_{\text{spectral}} = O\left(dn^2 + \text{polylog}(dN) + \log\left(\frac{\kappa}{\epsilon}\right)\right), \qquad N = 2^n, \tag{2.81}$$

where the QFTs contribute the dn^2 term and the reversible diagonal block encoding contributes the remaining terms. This bound concerns the coherent filter circuit; it assumes that $|\boldsymbol{b}_h\rangle$ is already available and does not include the cost of postselection, amplitude amplification, or final readout.

The postselection probability is

$$p_{\text{spec}} = \sum_{\boldsymbol{k}} |\rho_{\boldsymbol{k}}|^2 |\widehat{b}_{\boldsymbol{k}}|^2 = \frac{\|A_h^{-1}\boldsymbol{b}_h\|_2^2}{\alpha_{\text{inv}}^2 \|\boldsymbol{b}_h\|_2^2} \tag{2.82}$$

for normalized input. This formula is the simplest way to see how the condition number comes back. Since $\alpha_{\text{inv}} = \lambda_{\min}^{-1}$, one has $\rho_{\boldsymbol{k}} = \lambda_{\min}/\lambda_{\boldsymbol{k}}$. If $\boldsymbol{b}_h$ is concentrated in the highest-frequency eigenspace, then $p_{\text{spec}} = \Theta(\kappa^{-2})$, and amplitude amplification costs $\Theta(\kappa) = \Theta(h^{-2})$. If $\boldsymbol{b}_h$ is concentrated in the low modes, then p_{spec} can be $O(1)$.

This explains when filtering is useful. Smooth right-hand sides tend to have rapidly decaying high-frequency Fourier coefficients, and elliptic smoothing further suppresses high-frequency components in the solution. A spectral filter can exploit this favorable spectral measure, especially when the desired output is a low-frequency observable rather than the full normalized solution vector. But smoothness is not a substitute for an access model: one must still prepare the right-hand side state, compute the reciprocal accurately, and account for the postselection probability in (2.82).

The QFT route therefore does not generically produce the $\Theta(h^{-1})$ first-order scale of Section 2.2. It exploits a known eigenbasis rather than a mixed or factorized PDE system. Its great advantage is that, on separable constant-coefficient problems, the entire spectral action can be described by QFTs and reversible arithmetic. Its limitation is equally clear: variable coefficients, irregular domains, unstructured meshes, and general mixed boundary conditions destroy exact Fourier diagonalization.

2.4 Elliptic eigenvalue problems, mass lumping, and phase estimation

The same elliptic operator defines an eigenvalue problem. Because the weak form, stiffness matrix, mass matrix, and mass-normalized coordinates have already been introduced in Section 2.1.3, we only record the additional ingredients needed for eigenvalue approximation and quantum phase estimation.

2.4.1 Galerkin eigenvalues and their approximation

The continuous weak eigenproblem is: find $\lambda \in \mathbb{R}$ and $0 \neq \varphi \in H_0^1(\Omega)$ such that

$$a(\varphi, v) = \lambda(\varphi, v)_{L^2(\Omega)}, \qquad v \in H_0^1(\Omega). \tag{2.83}$$

The conforming finite element approximation uses the same space V_h, stiffness matrix K_h, and mass matrix M_h as in (2.27) and (2.30):

$$K_h \boldsymbol{y}_{j,h} = \lambda_{j,h} M_h \boldsymbol{y}_{j,h}. \tag{2.84}$$

The min–max principle immediately gives the one-sided Rayleigh–Ritz bound

$$\lambda_j \leq \lambda_{j,h}, \qquad 1 \leq j \leq \dim(V_h). \tag{2.85}$$

Thus conforming Galerkin projection overestimates the exact eigenvalues. For a fixed simple eigenpair with $\varphi_j \in H^2(\Omega)$, normalized in L^2, conforming P_1 elements satisfy [14]

$$0 \leq \lambda_{j,h} - \lambda_j \leq Ch^2, \qquad \|\varphi_j - \varphi_{j,h}\|_{H^1(\Omega)} \leq Ch, \qquad \|\varphi_j - \varphi_{j,h}\|_{L^2(\Omega)} \leq Ch^2. \tag{2.86}$$

More generally, eigenvalue errors are quadratic in the corresponding energy-norm approximation error.

For the one-dimensional finite-difference matrix, the discrete sine eigenpairs are already given by (2.13). For fixed mode index j and $L_{\mathrm{box}} = 1$,

$$\lambda_{j,h} = j^2\pi^2 - \frac{j^4\pi^4}{12}h^2 + O(j^6h^4). \tag{2.87}$$

Thus the low finite-difference eigenvalues are second-order accurate but, in contrast to conforming Galerkin eigenvalues, they underestimate the continuum eigenvalues. This sign difference is a useful reminder that finite differences and finite elements may share the same convergence order while preserving different variational inequalities.

2.4.2 Mass normalization and mass lumping

With the exact mass matrix, the generalized problem (2.84) becomes the standard Hermitian problem

$$A_h \boldsymbol{z}_{j,h} = \lambda_{j,h} \boldsymbol{z}_{j,h}, \qquad \boldsymbol{z}_{j,h} = M_h^{1/2} \boldsymbol{y}_{j,h}, \tag{2.88}$$

where A_h is the mass-normalized stiffness matrix from (2.32). The mathematical reduction is simple; implementing $M_h^{-1/2}$ can be nontrivial.

Mass lumping replaces M_h by a diagonal positive matrix M_h, often defined by row sums:

$$(M_h)_{ii} = \sum_j (M_h)_{ij}, \qquad (M_h)_{ij} = 0 \quad (i \neq j). \tag{2.89}$$

The resulting Hermitian matrix is

$$A_h = (M_h)^{-1/2} K_h (M_h)^{-1/2}. \tag{2.90}$$

Its diagonal scaling is easy to compute and preserves sparsity. Under standard assumptions, the low eigenvalues retain second-order accuracy for P_1 elements. However, because quadrature has changed the inner product, the strict Galerkin upper bound (2.85) need not hold for the lumped eigenvalues.

2.4.3 Phase estimation as spectral sampling

Quantum phase estimation is the circuit-level counterpart of expanding a vector in an eigenbasis. It is more than a subroutine for returning one eigenvalue. Its elegant feature is that it acts coherently on a superposition of eigenmodes. Suppose a block encoding or Hamiltonian simulation of A_h is available and prepare

$$|\boldsymbol{z}_0\rangle = \sum_j c_j |\boldsymbol{z}_{j,h}\rangle, \qquad A_h |\boldsymbol{z}_{j,h}\rangle = \lambda_{j,h} |\boldsymbol{z}_{j,h}\rangle. \tag{2.91}$$

Choose a simulation time t_0 and write

$$e^{-it_0 A_h} |\boldsymbol{z}_{j,h}\rangle = e^{-2\pi i \theta_j} |\boldsymbol{z}_{j,h}\rangle, \qquad \theta_j = \frac{t_0 \lambda_{j,h}}{2\pi} \pmod 1.$$

To recover the eigenvalue without aliasing, one chooses t_0 so that the relevant spectral interval fits inside one phase period; for example, $t_0 \lambda_{\max}(A_h) < 2\pi$ for the full spectrum. Since $\lambda_{\max}(A_h) = \Theta(h^{-2})$, this choice makes a single simulation step have length $t_0 = O(h^2)$ unless additional spectral information or rescaling is used. The no-aliasing condition is not the only cost. If adjacent eigenvalues are separated by $\Delta\lambda$, their QPE phases are separated by $t_0 \Delta\lambda/(2\pi)$. Since $t_0 = O(h^2)$ and low-frequency elliptic eigenvalue spacings are often $O(L_{\text{box}}^{-2})$, resolving individual low modes may require controlled powers of size $O(L_{\text{box}}^2/h^2)$. Thus high-resolution spectral sampling can reintroduce the elliptic gap scale, even though a single unaliased simulation step has length $O(h^2)$.

With r ancilla qubits, standard QPE applies controlled powers of this unitary and an inverse QFT on the phase register. Ideally, it performs the map

$$\sum_j c_j |\boldsymbol{z}_{j,h}\rangle |0^r\rangle \longmapsto \sum_j c_j |\boldsymbol{z}_{j,h}\rangle \left|\widetilde{\theta}_j\right\rangle, \tag{2.92}$$

where $\widetilde{\theta}_j$ is an r-bit approximation to θ_j. Measuring the phase register returns an approximate eigenvalue with probability close to $|c_j|^2$; leaving the register unmeasured produces an entangled state between eigenvectors and eigenvalue labels. This is the quantum analogue of spectral sampling. For the standard circuit, the number of phase qubits determines the resolution, and the controlled powers determine the total simulation time; see Nielsen and Chuang for the usual accuracy and success-probability estimates [71, Chapter 5].

Postselecting an eigenvalue window projects the spatial register onto the corresponding discrete eigenspace. This makes QPE useful for spectral projectors, low-mode filtering, and eigenstate preparation when the initial state has good overlap. It also reveals a limitation: QPE does not create the ground state from nothing. In simple Dirichlet problems, smooth positive trial functions may have useful overlap with the first eigenfunction.

There are many variants of QPE. Iterative and adaptive schemes reduce the number of coherent ancillas or change how measurements are processed; robust and low-depth variants are designed for early fault-tolerant regimes; and quantum signal processing gives another way to build spectral filters without explicitly storing an eigenvalue register. We do not need those refinements here, but they are important in modern resource estimates; see the survey-style discussion in [61] and the low-depth energy-estimation method of Ding and Lin [28].

The mesh scale remains visible. Since $\|A_h\| = \Theta(h^{-2})$, generic Hamiltonian simulation or phase estimation based directly on this operator inherits the normalization of the elliptic matrix unless QFT diagonalization, first-order factorization, or preconditioning is used. Generalized-eigenvalue phase-estimation methods offer another route when direct access to the matrix pair (K_h, M_h) is available [73].

2.5 What to remember

1. Standard second-order discretizations have L^2 error $O(h^2)$ under sufficient regularity, but their mass-normalized stiffness matrices have $\kappa(A_h) = \Theta(h^{-2})$.

2. A generic QSVT inverse filter therefore needs $\widetilde{O}(h^{-2})$ queries to a block encoding of A_h, even when each query has a compact sparse or structured circuit.

3. The factorization $A_h = G_h^\dagger G_h$ exposes the first-order scale $\kappa(G_h) = \Theta(h^{-1})$ on a fixed domain and can yield a square-root improvement under suitable access and range assumptions.

4. QFT-based arithmetic filters avoid the generic inverse-polynomial construction for separable constant-coefficient problems, but worst-case postselection can still retain $\Theta(h^{-2})$ dependence.

5. Conforming finite elements overestimate elliptic eigenvalues and give $O(h^2)$ eigenvalue errors for regular P_1 eigenfunctions. Mass lumping simplifies phase estimation but changes the variational problem.

6. Preconditioning, dynamics, stopping rules, and quantum-compatible finite element access are central to extending these introductory algorithms to realistic elliptic systems.

2.6 Outlook

The preceding sections illustrate three basic routes: direct inversion of A_h, first-order factorization, and explicit spectral diagonalization. Several broader issues remain important, but we discuss them only briefly.

Preconditioning. Classical elliptic solvers do not accept $\kappa(A_h) = \Theta(h^{-2})$ as unavoidable. Multigrid and BPX preconditioners use nested spaces to produce operators whose condition numbers are independent of the finest mesh size. A quantum preconditioner must itself admit an efficient block encoding and must not make state preparation or readout more expensive. The BPX-based construction of Deiml and Peterseim is an important example of treating multilevel structure as part of the quantum algorithm rather than as hidden classical preprocessing [15, 27].

Dynamics-based solvers. An elliptic solution can be realized as the steady state of a stable evolution, for example

$$\dot{\boldsymbol{x}}(t) = -B_h A_h \boldsymbol{x}(t) + B_h \boldsymbol{b}_h, \tag{2.93}$$

where B_h may be a preconditioner. Quantum ODE solvers, dilation methods, and Schrödingerization can then replace a direct inverse filter by simulation of the relaxation [47]. The useful complexity parameter is the stability and decay of the preconditioned evolution, not merely $\kappa(A_h)$.

Residual-based stopping. Classical iterative solvers stop when a residual is sufficiently small. A quantum dynamical solver can carry a residual register and estimate its weight without reconstructing the full solution [57]. This makes the runtime instance dependent and can exploit smooth right-hand sides whose high-frequency components decay rapidly.

Ground-state preparation. For elliptic eigenproblems, quantum imaginary-time evolution applies the filter

$$e^{-\tau A_h} |\boldsymbol{z}_0\rangle = \sum_j c_j e^{-\tau \lambda_{j,h}} |\boldsymbol{z}_{j,h}\rangle , \tag{2.94}$$

which suppresses excited modes after normalization. QITE and related dissipative methods address the overlap bottleneck left by phase estimation [70].

Systems of elliptic equations. Linear elasticity, coupled diffusion, and mixed formulations lead to vector-valued unknowns and block matrices. Strongly elliptic systems retain coercive energy estimates, whereas saddle-point formulations require inf–sup stability and suitable block preconditioners. Quantum algorithms must preserve these structural distinctions rather than treating every block matrix as a generic sparse system.

General finite element workflows. On unstructured meshes, efficient access should be formulated in terms of local elements, basis-function supports, quadrature data, coefficients, and boundary constraints. The remaining challenges include mass normalization, local-to-global indexing, variable coefficients, adaptive meshes, and observable extraction. These issues determine

whether a mathematically good finite element discretization also gives an efficient quantum access model.

Nonsymmetric elliptic problems. The discussion in this chapter has focused mainly on self-adjoint elliptic operators, for which the finite difference or finite element matrix is Hermitian positive definite. This structure is lost when the differential equation contains an advective term,

$$-\nabla \cdot (a(\boldsymbol{x})\nabla u) + \boldsymbol{\beta}(\boldsymbol{x}) \cdot \nabla u + c(\boldsymbol{x})u = f, \tag{2.95}$$

or when non-self-adjoint boundary conditions are imposed. Standard Dirichlet, Neumann, and real Robin conditions preserve symmetry for the diffusion operator, whereas oblique, impedance-type, or nonsymmetrically enforced boundary conditions may produce a nonsymmetric discretization. Upwind or streamline-stabilized treatments of (2.95) generally lead to

$$A_h = K_h + C_h, \qquad C_h \neq C_h^\dagger, \tag{2.96}$$

where K_h is the diffusion stiffness matrix and C_h represents advection, boundary, or stabilization terms.

For such matrices, the eigenvalue-transformation viewpoint used for Hermitian positive-definite systems no longer applies directly. One possible route is to introduce the Hermitian dilation

$$\mathcal{D}(A_h) = \begin{bmatrix} 0 & A_h^\dagger \\ A_h & 0 \end{bmatrix}, \tag{2.97}$$

whose nonzero eigenvalues are the signed singular values of A_h. QSVT may then implement a singular-value inverse filter, but it does not alleviate the $O(1/h^2)$ scaling of the condition number.

2.7 Exercises and further directions

Exercise 2.1 (One-dimensional factorization)**.** *Verify directly that $G_h^\dagger G_h$ is the matrix in (2.11). Compare the singular values of G_h with the square roots of (2.13).*

Exercise 2.2 (Two-dimensional Kronecker structure)**.** *Starting from (2.18), derive (2.19) and show that its eigenvectors are tensor products of one-dimensional sine modes. Then verify the stacked-gradient representation in (2.22).*

Exercise 2.3 (Mass normalization)**.** *For P_1 finite elements on a uniform one-dimensional mesh, write down K_h and M_h. Compare the spectra of K_h, $M_h^{-1}K_h$, and $M_h^{-1/2}K_hM_h^{-1/2}$.*

Exercise 2.4 (Direct QSVT scale)**.** *Use (2.14) to derive the normalized spectral interval (2.38) and the query estimate (2.39).*

Exercise 2.5 (Mixed first-order conditioning)**.** *Derive the two eigenvalues (2.51) of the mixed matrix $\mathsf{A}_h^{\mathrm{mix}}$ associated with a singular value σ of G_h. Show that if $\sigma_{\min} = \Theta(1)$, then $\kappa(\mathsf{A}_h^{\mathrm{mix}}) = \Theta(h^{-1})$, whereas if $\sigma_{\min} = \Theta(L_{\mathrm{box}}^{-1}) \ll 1$, the unscaled system has the larger condition number in (2.53).*

Exercise 2.6 (Galerkin eigenvalue bound)**.** *Use the min–max principle and $V_h \subset H_0^1(\Omega)$ to prove (2.85).*

Exercise 2.7 (Mass lumping)**.** *For P_1 elements in one dimension, compute the consistent and row-sum lumped mass matrices. Compare the resulting generalized eigenvalues with* (2.13).

Exercise 2.8 (Spectral-filter success probability)**.** *Evaluate* (2.82) *when $\boldsymbol{b}_h$ is a single low-frequency mode and when it is the highest-frequency mode. Explain the difference between arithmetic circuit depth and end-to-end state-preparation cost.*

Exercise 2.9 (Residual stopping)**.** *Assume $\|\boldsymbol{x}_* - \boldsymbol{x}(t)\| \le C\|\boldsymbol{r}(t)\|$ from equation* (2.93)*. If a normalized joint state contains solution and residual blocks, derive a bound on the relative error from the probability of measuring the residual block.*

Chapter 3

Quantum Algorithms for Hyperbolic PDEs

3.1 Classical wave discretizations: finite differences and finite elements

Hyperbolic equations model wave propagation rather than relaxation to static equilibrium. This is the main conceptual difference from the elliptic chapter. In an elliptic problem, the operator determines an equilibrium configuration. In contrast, in a hyperbolic problem, the same spatial operator generates a time evolution via Newton's second law.

Mathematically, the relation with Chapter 2 is direct. The stiffness operator in the elliptic model (2.1) becomes, after mass normalization, the spatial operator whose square root determines wave frequencies. The divergence–gradient factorization (2.35) therefore reappears below as the wave factorization (3.38). This is the first place where the distinction between an operator inverse problem and a unitary propagation problem becomes visible at the level of quantum circuits.

The time-dependent Schrödinger equation is itself commonly regarded as a wave equation because it propagates a complex wavefunction. Mathematically, however, it is a first-order dispersive evolution rather than a second-order hyperbolic equation. A central point of this chapter is that the classical second-order wave equation can nevertheless be rewritten, after an appropriate spatial discretization and a change to energy variables, as an ordinary Schrödinger equation with a Hermitian Hamiltonian. This paves the way to native quantum simulation of wave dynamics.

This observation also fixes the point of view used in the chapter. We do not try to reproduce a classical time-marching code gate by gate. Instead, we first build a semidiscrete Hamiltonian system from a standard finite difference or finite element approximation, and then use quantum Hamiltonian simulation to evolve that system. Thus the role of numerical analysis is to identify a stable, consistent, and physically meaningful finite-dimensional model, while the role of quantum algorithms is to implement the resulting unitary or nearly unitary evolution and to extract a limited number of quantities of interest. The distinction is important: a quantum computer is not an efficient printer of the full grid function, but it can be an efficient device for preparing and measuring structured states associated with the discrete wave energy.

The organizing principle of the chapter is energy conservation, and it is worth stating as a dial rather than a dichotomy. When the discrete dynamics conserves a Euclidean energy exactly, the quantum computer operates in its native mode: the evolution is unitary, there is no postselection penalty, and the only costs are simulation depth and measurement. Every departure from exact conservation—body forces, nonhomogeneous boundary data, absorbing layers, upwind viscosity—shows up as a specific, quantifiable quantum overhead: an LCU normalization, a success amplitude, or a dilation. The chapter is arranged along this dial. This section records the classical discretizations and their energy identities. Section 3.2 performs the main translation from the wave equation to a Schrödinger equation. Section 3.3 treats sources and boundary data by Duhamel's principle and LCU. Section 3.4 shows how explicit Fourier structure can fast-forward special cases, and Section 3.5 assembles the end-to-end pipeline for general meshes. Finally, Section 3.6 turns to first-order systems, where classical stabilization deliberately breaks conservation, and Schrödingerization, LCHS, and moment-matching dilation restore a unitary description on a larger space.

We begin with the scalar second-order wave equation

$$u_{tt}(\boldsymbol{x},t) - \nabla \cdot \left(c(\boldsymbol{x})^2 \nabla u(\boldsymbol{x},t)\right) = f(\boldsymbol{x},t), \qquad \boldsymbol{x} \in \Omega, \tag{3.1}$$

with initial conditions

$$u(\boldsymbol{x},0) = u_0(\boldsymbol{x}), \qquad u_t(\boldsymbol{x},0) = v_0(\boldsymbol{x}), \tag{3.2}$$

and, for the moment, homogeneous Dirichlet boundary condition $u|_{\partial\Omega} = 0$. For $f = 0$, the natural continuous energy is

$$E(t) = \frac{1}{2}\|u_t(t)\|^2_{L^2(\Omega)} + \frac{1}{2}\int_\Omega c(\boldsymbol{x})^2 |\nabla u(\boldsymbol{x},t)|^2 \, d\boldsymbol{x}. \tag{3.3}$$

Under the homogeneous boundary condition, integration by parts gives $dE/dt = 0$. This identity is the main guide for both the classical discretization and the quantum construction. A good discretization should not merely approximate u_{tt} and ∇u. Instead, it should preserve, or at least respect, the underlying energy structure.

There are two useful ways to rewrite the second-order equation as a first-order system. The first uses the displacement–velocity pair. With

$$v = u_t, \qquad \boldsymbol{z} = \begin{bmatrix} u \\ v \end{bmatrix}, \tag{3.4}$$

the homogeneous equation has the formal representation

$$\frac{d}{dt}\boldsymbol{z} = \begin{bmatrix} 0 & I \\ -L & 0 \end{bmatrix} \boldsymbol{z}, \qquad L = -\nabla \cdot (c^2 \nabla). \tag{3.5}$$

This form keeps the displacement explicit and is convenient when the desired output is a displacement observable. The second formulation uses a pressure, strain, or deformation-gradient variable together with velocity. We write

$$\boldsymbol{p} = c\nabla u, \qquad v = u_t. \tag{3.6}$$

For the homogeneous equation,

$$\boldsymbol{p}_t = c\nabla v, \qquad v_t = \nabla \cdot (c\boldsymbol{p}). \tag{3.7}$$

This pressure–velocity form is closer to the conserved energy (3.3). It is especially natural when the output is elastic or acoustic energy, strain energy, flux, or modal energy. Throughout the chapter we use $\boldsymbol{p}_h = G_h \boldsymbol{y}_h$ for the discrete pressure/strain block and $\boldsymbol{v}_h = \dot{\boldsymbol{y}}_h$ for the discrete velocity block. The main Hamiltonian formulation below is based on these variables, while Subsection 3.5.1 explains how to carry the displacement as an additional block when needed.

For numerical analysts, this is just the familiar passage from a second-order equation to a first-order system in energy variables. For quantum computing, it has an additional meaning. The pressure–velocity variables are exactly the variables whose Euclidean norm is conserved by the semidiscrete dynamics. They are therefore the natural amplitudes of a quantum state. If one used the raw displacement vector instead, the conserved norm would involve the stiffness matrix and mass matrix, and the corresponding generator would not be Hermitian in the standard amplitude inner product.

3.1.1 Centered finite differences

As examples of classical algorithms, we again start with a finite difference scheme. On the interval $\Omega = (0, L_{\text{box}})$, with c constant and $f = 0$, consider

$$u_{tt} = c^2 u_{xx}, \qquad u(0,t) = u(L_{\text{box}}, t) = 0. \tag{3.8}$$

For a simple finite difference approximation for this model, we use the same grid points (2.8) with N_h interior points and spacing h. The standard centered fully discrete scheme in both space and time is

$$\frac{U_j^{n+1} - 2U_j^n + U_j^{n-1}}{k^2} = c^2 \frac{U_{j+1}^n - 2U_j^n + U_{j-1}^n}{h^2}, \qquad j = 1, \dots, N_h. \tag{3.9}$$

Here U_j^n approximates $u(x_j, t_n)$, and

$$\boldsymbol{u}_h^n = (U_1^n, \dots, U_{N_h}^n)^T \tag{3.10}$$

denotes the vector of interior nodal values.

Because (3.9) is second order in time, it requires two starting vectors. The first is obtained directly from the displacement data,

$$\boldsymbol{u}_h^0 \approx (u_0(x_1), \dots, u_0(x_{N_h}))^T.$$

The second initial vector is chosen by a Taylor expansion using the initial velocity. For $\ddot{\boldsymbol{u}}_h = -A_h \boldsymbol{u}_h + \boldsymbol{f}_h(t)$, a second-order initialization is

$$\boldsymbol{u}_h^1 = \boldsymbol{u}_h^0 + k\boldsymbol{v}_h^0 - \frac{k^2}{2} A_h \boldsymbol{u}_h^0 + \frac{k^2}{2} \boldsymbol{f}_h(0), \tag{3.11}$$

with $\boldsymbol{v}_h^0$ obtained from $v_0 = u_t(x, 0)$. In the homogeneous constant-speed case, this is the familiar formula obtained by replacing $u_{tt}(0)$ with $c^2 u_{xx}(0)$. A lower-order initialization would contaminate the global second-order accuracy, which is why this first step is part of the classical scheme rather than a harmless implementation detail.

We use the weighted discrete norm

$$\|\boldsymbol{w}\|_{\ell_h^2} := \left(h \sum_{j=1}^{N_h} |w_j|^2 \right)^{1/2}, \tag{3.12}$$

which is the one-dimensional version of the $h^{d/2}$-scaling between L^2 functions and Euclidean amplitude vectors discussed in Chapter 1.

The error analysis of a finite difference approximation like (3.9) often begins with the local error. A Taylor expansion of the fully discrete formula, after substitution of the exact solution, gives the local residual

$$\begin{aligned} \tau_j^n := & \left[u_{tt}(x_j, t_n) - \frac{u(x_j, t_{n+1}) - 2u(x_j, t_n) + u(x_j, t_{n-1})}{k^2} \right] \\ & - c^2 \left[u_{xx}(x_j, t_n) - \frac{u(x_{j+1}, t_n) - 2u(x_j, t_n) + u(x_{j-1}, t_n)}{h^2} \right] = O(k^2 + h^2). \end{aligned} \tag{3.13}$$

This displayed form is longer than the original difference equation, but it makes the two sources of second-order consistency visible: the centered time difference and the centered spatial difference.

Consistency alone does not imply convergence. For the wave equation, stability requires the Courant–Friedrichs–Lewy restriction

$$\nu := \frac{ck}{h} \le 1. \tag{3.14}$$

For a uniform stability margin $\nu \le \nu_0 < 1$, and sufficiently smooth exact solutions, a representative estimate is

$$\max_{0 \le n \le T/k} \|\boldsymbol{u}_h^n - \boldsymbol{u}_{\mathrm{ex}}^n\|_{\ell_h^2} \le C(1+T)(h^2 + k^2), \qquad \boldsymbol{u}_{\mathrm{ex}}^n = \big(u(x_1, t_n), \ldots, u(x_{N_h}, t_n)\big)^T, \tag{3.15}$$

where the constant depends on a finite number of derivatives of u, but not on h, k, or T [53]. The factor $1 + T$ is important: wave equations transport consistency errors rather than smoothing them.

This should be contrasted with the heat equation in Chapter 4. Parabolic smoothing damps high-frequency local errors, whereas the wave equation carries local errors along characteristics and reflects them from boundaries. This is why the classical CFL time step and the quantum Hamiltonian-simulation normalization both contain the scale h^{-1}: the fastest numerical waves cross one grid cell in time $O(h)$.

The centered scheme in (3.9) is useful for explaining classical accuracy and the CFL restriction. The quantum algorithm developed below does *not* have to apply this classical time step coherently. Instead, we typically discretize only in space and retain continuous time. Define

$$\boldsymbol{u}_h(t) = (U_1(t), \ldots, U_{N_h}(t))^T. \tag{3.16}$$

For constant c, the semidiscrete equation is

$$\ddot{\boldsymbol{u}}_h(t) + A_h \boldsymbol{u}_h(t) = \boldsymbol{0}, \qquad A_h = \frac{c^2}{h^2} \begin{bmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{bmatrix}. \tag{3.17}$$

Thus A_h is c^2 times the one-dimensional elliptic matrix in (2.11). It is symmetric positive definite, and

$$\|A_h^{1/2}\| = \Theta(h^{-1}). \tag{3.18}$$

This is the hyperbolic analogue of the first-order scale $\|G_h\| = \Theta(h^{-1})$ encountered in Chapter 2.

The same construction extends naturally to variable wave speed. Toward this end, we write

$$a(x) = c(x)^2, \qquad -\partial_x(a(x)\partial_x u) \tag{3.19}$$

for the positive spatial operator. Then the natural discretization is obtained by first approximating the edge flux

$$F_{j+1/2}(t) \approx a_{j+1/2}\frac{U_{j+1}(t) - U_j(t)}{h}, \qquad a_{j+1/2} = a(x_{j+1/2}), \tag{3.20}$$

and then differencing the fluxes. This gives

$$(A_h \boldsymbol{u}_h)_j = \frac{1}{h^2}\left[a_{j+1/2}(U_j - U_{j+1}) + a_{j-1/2}(U_j - U_{j-1})\right]. \tag{3.21}$$

Thus A_h remains symmetric. Its diagonal entries are $(a_{j-1/2} + a_{j+1/2})/h^2$, and its off-diagonal entries are $-a_{j+1/2}/h^2$. This is the wave-equation counterpart of the variable-coefficient elliptic stiffness matrix from Chapter 2. The conservative flux form is also the intuitive reason for the symmetry: the same edge coefficient $a_{j+1/2}$ couples the two neighboring nodes j and $j+1$.

This edge-centered viewpoint is also the most useful one for quantum access. A sparse-matrix oracle does not need to store the full matrix; it only needs to compute the neighboring indices and the local coefficients $a_{j\pm1/2}$. This is analogous to the sparse access model in Section 1.11.3: the finite difference stencil supplies the locations, while the coefficient field supplies the values. For smooth or piecewise smooth $c(x)$, these values can often be computed reversibly from the grid index.

In several dimensions, we again keep the time variable continuous and write the spatially semidiscrete equation in the same form as

$$\ddot{\boldsymbol{u}}_h(t) + A_h \boldsymbol{u}_h(t) = \boldsymbol{f}_h(t), \tag{3.22}$$

where A_h is the multi-dimensional centered difference matrix. On Cartesian grids, A_h is assembled from the same directional second differences used in the five-point elliptic operator (2.18); for variable coefficients it is assembled from directional edge fluxes. It has $O(d)$ nonzero entries per row and $\|A_h^{1/2}\| = \Theta(h^{-1})$ under uniform ellipticity of $c(\boldsymbol{x})^2$. Higher-order and extended stencils lead to the same semidiscrete form, but their factorizations can be more delicate; see Subsection 3.2.1.

3.1.2 Finite elements and the mass matrix

Finite elements begin from the same bilinear form and stiffness matrix used for the elliptic problem in Section 2.1.3. For homogeneous Dirichlet boundary conditions, define

$$a(u, v) = \int_\Omega c(\boldsymbol{x})^2 \nabla u(\boldsymbol{x}) \cdot \nabla v(\boldsymbol{x})\, d\boldsymbol{x}. \tag{3.23}$$

The variational wave equation is: find $u(\cdot,t) \in H_0^1(\Omega)$ such that

$$(u_{tt}(\cdot,t), v) + a(u(\cdot,t), v) = (f(t), v), \qquad v \in H_0^1(\Omega). \tag{3.24}$$

Let $V_h \subset H_0^1(\Omega)$ be a conforming finite element space with basis $\{\phi_1, \dots, \phi_{N_h}\}$, and write

$$u_h(\boldsymbol{x}, t) = \sum_{j=1}^{N_h} (\boldsymbol{q}_h(t))_j \phi_j(\boldsymbol{x}). \tag{3.25}$$

The semidiscrete method, after enforcing the weak form, is given by,

$$M_h \ddot{\boldsymbol{q}}_h(t) + K_h \boldsymbol{q}_h(t) = \boldsymbol{b}_h(t), \tag{3.26}$$

where

$$(M_h)_{ij} = (\phi_j, \phi_i), \qquad (K_h)_{ij} = a(\phi_j, \phi_i), \qquad (\boldsymbol{b}_h(t))_i = (f(t), \phi_i). \tag{3.27}$$

Here $(\cdot,\cdot)$ refers to the L^2 inner product.

The stiffness and mass matrices are precisely the matrices introduced in (2.28) and (2.30), now appearing in a time-dependent equation. Under homogeneous Dirichlet conditions, M_h and K_h are symmetric positive definite. For $f = 0$, multiplication by $\dot{\boldsymbol{q}}_h(t)^\dagger$ gives

$$\frac{d}{dt}\left[\frac{1}{2}\dot{\boldsymbol{q}}_h(t)^\dagger M_h \dot{\boldsymbol{q}}_h(t) + \frac{1}{2}\boldsymbol{q}_h(t)^\dagger K_h \boldsymbol{q}_h(t)\right] = 0. \tag{3.28}$$

Thus the finite element semidiscretization preserves the continuous energy structure automatically. From the quantum point of view, this identity also tells us which vector should be normalized. The Euclidean norm of the coefficient vector $\boldsymbol{q}_h$ is not the physical L^2 norm. The physical kinetic energy uses M_h, while the strain energy uses K_h. This is why the mass-normalized variable $\boldsymbol{y}_h = M_h^{1/2}\boldsymbol{q}_h$ will appear repeatedly below.

For piecewise linear elements on a quasi-uniform mesh, compatible projections of the initial data, and a sufficiently smooth solution, representative estimates are

$$\|u_h(t) - u(t)\|_{L^2(\Omega)} \le Ch^2 \left(\|u(t)\|_{H^2(\Omega)} + \int_0^t \|u_{tt}(s)\|_{H^2(\Omega)}\, ds\right), \tag{3.29}$$

$$\|\nabla(u_h(t) - u(t))\|_{L^2(\Omega)} \le Ch \left(\|u(t)\|_{H^2(\Omega)} + \int_0^t \|u_{tt}(s)\|_{H^1(\Omega)}\, ds\right), \tag{3.30}$$

up to the usual initial projection terms. The integrals over $[0,t]$ again express the accumulation of consistency error in a nonsmoothing evolution. Classical optimal-order analyses include the work of Dupont, Baker, Baker and Bramble, Dougalis, and Baker–Dougalis–Serbin [30, 8, 9, 29, 10].

The finite element formulation is useful pedagogically because it separates three issues that are often merged in a finite difference formula. The matrix K_h represents the elastic or acoustic restoring force, the matrix M_h represents the discrete L^2 inner product, and the energy identity tells us which combination of position and velocity is conserved. Quantum amplitude encoding uses the ordinary Euclidean norm on coefficient vectors, so the mass matrix cannot be ignored. The mass normalization in (3.35) is precisely the finite element analogue of the L^2-to-ℓ_2 scaling discussed in Chapter 1.

3.1.3 Mass lumping

The finite element mass matrix M_h is sparse, but $M_h^{-1/2}$ is generally dense. In principle, if a normalized block encoding of M_h is available, QSVT can approximate the scalar map $x \mapsto x^{-1/2}$ on the spectral interval of M_h and thereby produce a block encoding of $M_h^{-1/2}$. This is a legitimate route, especially because the condition number of a properly scaled mass matrix is mesh independent on a shape-regular quasi-uniform mesh. It nevertheless requires an additional spectral transformation and produces a nonlocal operator at the matrix level.

For an introductory construction, mass lumping is much simpler. Replace M_h by a positive diagonal approximation

$$M_{h,L} = \operatorname{diag}(m_1, \dots, m_{N_h}), \qquad m_i > 0, \tag{3.31}$$

obtained, for example, by nodal quadrature or row summation. The lumped system is

$$M_{h,L}\ddot{\boldsymbol{q}}_h(t) + K_h \boldsymbol{q}_h(t) = \boldsymbol{b}_h(t), \tag{3.32}$$

and the normalized stiffness

$$A_{h,L} = M_{h,L}^{-1/2} K_h M_{h,L}^{-1/2} \tag{3.33}$$

retains the locality of K_h. This is the same mass-lumping mechanism used for the elliptic eigenproblem in Subsection 2.4.2.

From the quantum perspective, the main advantage is not that lumping is a better finite element approximation; it is that it keeps the transformation to Euclidean coordinates local. The diagonal entries $m_i^{-1/2}$ can be applied by reversible arithmetic or by a simple diagonal block encoding, while a consistent mass matrix would require an additional matrix-function primitive. Thus mass lumping trades a small classical quadrature modification for a much simpler quantum access model.

The formulas above used homogeneous boundary conditions and, for most of the discussion, $f = 0$. This is the cleanest starting point because the homogeneous wave equation becomes unitary after the energy-variable transformation. Body forces and nonhomogeneous boundary data can be incorporated by combining the homogeneous propagator with the variation-of-constants formula. We give the quantum formulation in Section 3.3.

3.2 From the wave equation to a Schrödinger equation

This section performs the main translation step of the chapter. The continuous PDE is first replaced by a finite-dimensional conservative mechanical system. We then choose coordinates in which the conserved energy is the Euclidean norm. Only after these two numerical-analysis steps does the system become a genuine Schrödinger equation, which is then directly suitable for Hamiltonian simulation. Both steps are familiar operations—semidiscretization and a change to energy variables—but the quantum setting sharpens the second one: it is not enough for the conserved quantity to be *equivalent* to the Euclidean norm, as in a standard stability argument; it must *be* the Euclidean norm, exactly, because that is the norm a quantum state carries.

The homogeneous semidiscrete wave equation has the form

$$M_h\ddot{\boldsymbol{q}}_h(t) + K_h\boldsymbol{q}_h(t) = \mathbf{0}, \qquad M_h = M_h^\dagger \succ 0, \qquad K_h = K_h^\dagger \succeq 0. \tag{3.34}$$

Assume first that the boundary conditions remove the nullspace, so that $K_h \succ 0$. Introduce the mass-normalized displacement

$$\boldsymbol{y}_h(t) = M_h^{1/2}\boldsymbol{q}_h(t), \qquad A_h = M_h^{-1/2}K_hM_h^{-1/2}. \tag{3.35}$$

Then

$$\ddot{\boldsymbol{y}}_h(t) + A_h\boldsymbol{y}_h(t) = \mathbf{0}, \qquad A_h = A_h^\dagger \succ 0. \tag{3.36}$$

This is the dynamical counterpart of the mass-normalized elliptic system (2.32).

The direct displacement–velocity system is

$$\frac{d}{dt}\begin{bmatrix}\boldsymbol{y}_h\\ \dot{\boldsymbol{y}}_h\end{bmatrix} = \begin{bmatrix}0 & I\\ -A_h & 0\end{bmatrix}\begin{bmatrix}\boldsymbol{y}_h\\ \dot{\boldsymbol{y}}_h\end{bmatrix}. \tag{3.37}$$

Although this system preserves the physical energy, its coefficient matrix is not skew-Hermitian in the ordinary Euclidean inner product. It is Hamiltonian in the classical symplectic sense, but a quantum computer simulates unitary evolution in a Hilbert-space inner product. We therefore change variables so that the conserved energy becomes an ordinary Euclidean norm. A direct way to do this is to use a square-root factorization

$$A_h = G_h^\dagger G_h. \tag{3.38}$$

For finite differences, G_h is a scaled discrete gradient. For finite elements, it is the mass-normalized gradient factor associated with (2.35). Define the pressure/strain block and the velocity block by

$$\boldsymbol{p}_h(t) = G_h\boldsymbol{y}_h(t), \qquad \boldsymbol{v}_h(t) = \dot{\boldsymbol{y}}_h(t). \tag{3.39}$$

Then

$$\dot{\boldsymbol{p}}_h(t) = G_h\boldsymbol{v}_h(t), \qquad \dot{\boldsymbol{v}}_h(t) = -G_h^\dagger\boldsymbol{p}_h(t). \tag{3.40}$$

Equivalently,

$$\frac{d}{dt}\boldsymbol{\psi}_h(t) = S_h\boldsymbol{\psi}_h(t), \qquad \boldsymbol{\psi}_h(t) = \begin{bmatrix}\boldsymbol{p}_h(t)\\ \boldsymbol{v}_h(t)\end{bmatrix}, \qquad S_h = \begin{bmatrix}0 & G_h\\ -G_h^\dagger & 0\end{bmatrix}. \tag{3.41}$$

Since $S_h^\dagger = -S_h$, multiplication by i gives the Schrödinger equation

$$i\frac{d}{dt}\boldsymbol{\psi}_h(t) = H_h\boldsymbol{\psi}_h(t), \qquad H_h = iS_h = i\begin{bmatrix}0 & G_h\\ -G_h^\dagger & 0\end{bmatrix}, \qquad H_h = H_h^\dagger. \tag{3.42}$$

This factorized Hamiltonian is the structural basis of the wave-equation algorithm of Costa, Jordan, and Ostrander [24], where finite difference forms were considered. Its norm is tied to the energy:

$$\|\boldsymbol{\psi}_h(t)\|_2^2 = \|G_h\boldsymbol{y}_h(t)\|_2^2 + \|\dot{\boldsymbol{y}}_h(t)\|_2^2 = \boldsymbol{y}_h(t)^\dagger A_h\boldsymbol{y}_h(t) + \|\dot{\boldsymbol{y}}_h(t)\|_2^2. \tag{3.43}$$

Thus the quantum state naturally represents the pressure–velocity, or energy, variables rather than the displacement alone. This is often an advantage: modal energy, local flux, kinetic energy, and strain energy are all functions of $G_h\boldsymbol{y}_h$ and $\dot{\boldsymbol{y}}_h$.

Notice the contrast with the elliptic and parabolic chapters. In an elliptic problem, the main primitive is an inverse or a resolvent; in a parabolic problem, it is a contraction semigroup. Here the homogeneous evolution is unitary after the correct change of variables. Consequently, there is no final postselection penalty associated with the norm of the propagated state. The measurement bottleneck instead comes from the usual need to estimate observables from repeated state preparations.

3.2.1 Finite differences as a factored system

For the one-dimensional Dirichlet discretization, define the first-difference matrix

$$D_h = \frac{1}{h}\begin{bmatrix} 1 & 0 & \cdots & 0 \\ -1 & 1 & \ddots & \vdots \\ & \ddots & \ddots & 0 \\ 0 & \cdots & -1 & 1 \\ 0 & \cdots & 0 & -1 \end{bmatrix}. \tag{3.44}$$

It maps N_h interior nodal values to $N_h + 1$ edge differences, including the two boundary edges. For constant c,

$$A_h = c^2 D_h^\dagger D_h, \qquad G_h = cD_h. \tag{3.45}$$

For variable coefficient $a(x) = c(x)^2$, let W_h be the positive diagonal matrix of edge coefficients. Then

$$A_h = D_h^\dagger W_h D_h, \qquad G_h = W_h^{1/2} D_h, \tag{3.46}$$

which is the factorized form of the conservative stencil (3.21). Notice that the stiffness matrix itself has entries of size $O(h^{-2})$, while the factor G_h has entries of size $O(h^{-1})$. The wave Hamiltonian uses the first-order factor, not the stiffness matrix directly. This is why the generic hyperbolic scale is T/h, in contrast to the direct elliptic inverse scale h^{-2} discussed in Chapter 2. The corresponding Hamiltonian is sparse, its entries are $O(h^{-1})$, and

$$\|H_h\| = \|G_h\| = \Theta(h^{-1}). \tag{3.47}$$

In higher dimensions, G_h is obtained by stacking directional difference matrices. In two dimensions, for example,

$$G_h = c\begin{bmatrix} D_x \\ D_y \end{bmatrix}, \qquad G_h^\dagger G_h = c^2(D_x^\dagger D_x + D_y^\dagger D_y) = A_h. \tag{3.48}$$

A simple example of a wider stencil is a one-dimensional lattice with both nearest-neighbor and next-nearest-neighbor couplings. On a periodic grid, let

$$(D_1 \boldsymbol{u})_j = \frac{U_{j+1} - U_j}{h}, \qquad (D_2 \boldsymbol{u})_j = \frac{U_{j+2} - U_j}{h}. \tag{3.49}$$

This is also the ideas in [24]. If the two spring constants η_1, η_2 are both nonnegative, then

$$A_h = \eta_1 D_1^\dagger D_1 + \eta_2 D_2^\dagger D_2 = G_h^\dagger G_h, \qquad G_h = \begin{bmatrix} \sqrt{\eta_1} D_1 \\ \sqrt{\eta_2} D_2 \end{bmatrix}. \tag{3.50}$$

This gives a five-point stencil whose Fourier symbol is

$$\lambda(\theta) = \frac{4\eta_1}{h^2}\sin^2\frac{\theta}{2} + \frac{4\eta_2}{h^2}\sin^2\theta. \tag{3.51}$$

The more interesting lattice-dynamics point in [56] is that the next-nearest-neighbor coefficient need not be positive term by term. What is needed for a stable wave equation is nonnegativity of the total symbol. For example, take $\eta_2 < 0$. Since

$$\sin^2\theta = 4\sin^2(\theta/2)\cos^2(\theta/2),$$

the symbol in (3.51) is nonnegative provided

$$\eta_1 + 4\eta_2 \geq 0. \tag{3.52}$$

The stacked construction (3.50) is now unavailable, because $\sqrt{\eta_2}$ is imaginary. The question becomes whether the nonnegative symbol can still be written as $|g(\theta)|^2$ for a *local* difference operator—that is, whether the factorization can be carried out without leaving the space of short stencils. It can. Readers may recognize what follows as a small instance of the Fejér–Riesz factorization of nonnegative trigonometric polynomials. Choose real numbers α, β such that

$$\alpha\beta = \eta_2, \qquad \alpha^2 + \beta^2 = \eta_1 + 2\eta_2. \tag{3.53}$$

These two conditions simply match the constant and the $\cos\theta$ coefficients of the two trigonometric polynomials, and such α, β exist under (3.52). Define

$$(G_h\boldsymbol{u})_j = \frac{1}{h}\left[\alpha(U_{j+1} - U_j) + \beta(U_{j+2} - U_{j+1})\right]. \tag{3.54}$$

The Fourier symbol of G_h is

$$g(\theta) = \frac{1}{h}(e^{i\theta} - 1)(\alpha + \beta e^{i\theta}),$$

and therefore

$$|g(\theta)|^2 = \frac{4\eta_1}{h^2}\sin^2\frac{\theta}{2} + \frac{4\eta_2}{h^2}\sin^2\theta. \tag{3.55}$$

As a concrete stable choice with a negative next-nearest-neighbor coefficient, take $\eta_1 = 5$, $\eta_2 = -1$. Then $\eta_1 + 4\eta_2 = 1 > 0$, and one may choose $\alpha = (1+\sqrt{5})/2$, $\beta = -(\sqrt{5}-1)/2$. This example illustrates why the factor G_h, rather than the individual stencil coefficients, is the quantum object of interest. Even a wider stencil with some negative couplings can lead to a Hermitian wave Hamiltonian once its nonnegative symbol has been factored as $A_h = G_h^\dagger G_h$.

3.2.2 Generic Hamiltonian-simulation complexity

Suppose that G_h/α_G has an efficient block encoding with

$$\alpha_G = \Theta(\|G_h\|) = \Theta(h^{-1}). \tag{3.56}$$

Then H_h/α_G has a block encoding with comparable cost. Indeed, the Hermitian dilation from Chapter 2,

$$\mathcal{D}(G_h) = \begin{bmatrix} 0 & G_h^\dagger \\ G_h & 0 \end{bmatrix},$$

is converted to $H_h = i\begin{bmatrix} 0 & G_h \\ -G_h^\dagger & 0 \end{bmatrix}$ by a phase on the block label, equivalently by conjugating with $\mathrm{diag}(I, iI)$. Thus the dilation block encoding gives the wave Hamiltonian block encoding using one extra label qubit and only constant overhead. Qubitization or QSVT Hamiltonian simulation implements e^{-iH_hT} with query complexity

$$\widetilde{O}\left(\alpha_G T + \log\frac{1}{\epsilon}\right) = \widetilde{O}\left(\frac{T}{h} + \log\frac{1}{\epsilon}\right). \tag{3.57}$$

Thus the best generic mesh-dependent scaling exposed by the first-order formulation is linear in T/h, rather than T/h^2. Up to logarithmic factors, this linear dependence on the scaled evolution time is optimal in the black-box setting: the no-fast-forwarding theorem rules out sublinear-time simulation for general sparse Hamiltonians [12, 67]; see also the discussion in Childs' lecture notes [21]. Any improvement beyond T/h must therefore exploit additional structure, such as the explicitly computable dispersion relation used in Section 3.4.

It is useful to compare this scale with a classical explicit method while keeping the physical parameters visible. Let L_{box} be a characteristic side length of the computational domain, let h be the mesh size, and let $c_{\max}$ be the maximum wave speed. On a uniform grid,

$$N_h \asymp \left(\frac{L_{\mathrm{box}}}{h}\right)^d$$

spatial degrees of freedom are used. A stable leapfrog computation from equation (3.9) satisfies a CFL restriction of the form

$$k \le C_{\mathrm{CFL}}\frac{h}{c_{\max}},$$

where the constant depends on the spatial dimension and the stencil but not on h, T, or L_{box}. Hence the number of time steps is

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

Since each sparse time step costs $O(N_h)$ arithmetic operations, the classical explicit work is

$$O(N_tN_h) = O\left(\frac{c_{\max}T}{h}\left(\frac{L_{\mathrm{box}}}{h}\right)^d\right).$$

The quantum Hamiltonian-simulation cost for the energy-state formulation is instead governed by the largest wave frequency,

$$\|H_h\|T \asymp \frac{c_{\max}T}{h}.$$

Thus the same CFL scale $c_{\max}/h$ appears, but without the extra factor counting all grid degrees of freedom. The domain size L_{box} enters the quantum algorithm through the number of qubits, state preparation, and readout model, rather than through the Hamiltonian-simulation time scale itself.

This is different from the elliptic chapter. For elliptic equations, solving $Lu = f$ requires applying an inverse, and the smallest eigenvalue scales like L_{box}^{-2}. Therefore the elliptic condition number contains $(L_{\mathrm{box}}/h)^2$, or L_{box}/h after first-order factorization. For the wave equation,

forward simulation is controlled by the largest frequency, which scales like $c_{\max}/h$, not by the low-frequency gap. The low-frequency scale would reappear only if one tries to invert the wave operator, recover displacement from strain by a pseudo-inverse, or perform high-resolution spectral estimation near zero frequency.

The quantum cost above reflects only coherent evolution of an amplitude-encoded energy state. It does not include input preparation or readout of all N_h grid values. This is why the most meaningful quantum outputs are low-dimensional observables, overlaps, modal weights, or sampled energy distributions rather than the full wave field.

3.3 Body forces and nonhomogeneous boundary conditions

A common question from numerical analysts is what happens when the wave equation has a body force or nonzero boundary data. For the wave equation this can be formulated cleanly, because the homogeneous part is already a Hamiltonian-simulation problem. The nonhomogeneous term enters through variation of constants. After a quadrature approximation for the time integral, the result can be implemented by a controlled superposition of homogeneous propagators and source-state preparations, closely related to the inhomogeneous constructions used in LCU and linear-combination-of-Hamiltonian-simulations (LCHS) algorithms [4]. In the language of the conservation dial from the chapter introduction, this section takes the first step away from exact conservation: the dynamics is still built entirely from unitary propagators, but a coefficient bookkeeping—an LCU normalization—now enters the cost.

Start from

$$M_h \ddot{\boldsymbol{q}}_h(t) + K_h \boldsymbol{q}_h(t) = \boldsymbol{b}_h(t). \tag{3.58}$$

After the same mass normalization,

$$\boldsymbol{y}_h = M_h^{1/2} \boldsymbol{q}_h, \qquad A_h = M_h^{-1/2} K_h M_h^{-1/2}, \qquad \boldsymbol{f}_h(t) = M_h^{-1/2} \boldsymbol{b}_h(t), \tag{3.59}$$

we obtain

$$\ddot{\boldsymbol{y}}_h(t) + A_h \boldsymbol{y}_h(t) = \boldsymbol{f}_h(t). \tag{3.60}$$

Assume $A_h = G_h^\dagger G_h$ and define

$$\boldsymbol{\psi}_h(t) = \begin{bmatrix} \boldsymbol{p}_h(t) \\ \boldsymbol{v}_h(t) \end{bmatrix}, \qquad \boldsymbol{p}_h = G_h \boldsymbol{y}_h, \qquad \boldsymbol{v}_h = \dot{\boldsymbol{y}}_h. \tag{3.61}$$

Then

$$\dot{\boldsymbol{\psi}}_h(t) = S_h \boldsymbol{\psi}_h(t) + \boldsymbol{g}_h(t), \qquad S_h = \begin{bmatrix} 0 & G_h \\ -G_h^\dagger & 0 \end{bmatrix}, \qquad \boldsymbol{g}_h(t) = \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{f}_h(t) \end{bmatrix}. \tag{3.62}$$

With $H_h = iS_h$, this is

$$i\dot{\boldsymbol{\psi}}_h(t) = H_h \boldsymbol{\psi}_h(t) + i\boldsymbol{g}_h(t), \tag{3.63}$$

and the homogeneous propagator is

$$U_h(\tau) = e^{-iH_h\tau} = e^{S_h\tau}. \tag{3.64}$$

Variation of constants gives

$$\boldsymbol{\psi}_h(T) = U_h(T)\boldsymbol{\psi}_h(0) + \int_0^T U_h(T-s)\boldsymbol{g}_h(s)\,ds. \tag{3.65}$$

The same Hamiltonian-simulation circuit used for the homogeneous equation is therefore reused at every quadrature node.

At the PDE level, (3.65) says that the final wave is a superposition of waves emitted by the source at earlier times. A numerical quadrature rule turns this continuum superposition into a finite sum. A quantum LCU implementation mirrors this interpretation: the clock or quadrature register chooses the emission time, the source oracle prepares the emitted wave packet, and the homogeneous propagator transports it from that time to T.

A quadrature rule with nodes s_m and weights w_m gives

$$\boldsymbol{\psi}_{h,M}(T) := U_h(T)\boldsymbol{\psi}_h(0) + \sum_{m=1}^{M_q} w_m U_h(T-s_m)\boldsymbol{g}_h(s_m) \approx \boldsymbol{\psi}_h(T). \tag{3.66}$$

To see explicitly how this becomes an LCU, define the normalized source states

$$|\widehat{\boldsymbol{g}}_m\rangle = \frac{\boldsymbol{g}_h(s_m)}{\|\boldsymbol{g}_h(s_m)\|}, \qquad \left|\widehat{\boldsymbol{\psi}}_0\right\rangle = \frac{\boldsymbol{\psi}_h(0)}{\|\boldsymbol{\psi}_h(0)\|}. \tag{3.67}$$

The magnitudes $\|\boldsymbol{g}_h(s_m)\|$ are not part of the normalized quantum state; they must be placed in the LCU coefficients. Introduce

$$\Lambda = \|\boldsymbol{\psi}_h(0)\| + \sum_{m=1}^{M_q} |w_m|\,\|\boldsymbol{g}_h(s_m)\|. \tag{3.68}$$

This is the same normalization parameter that appears in the LCU construction of Chapter 1: it is the sum of the absolute coefficients of the terms being coherently added.

A convenient PREPARE oracle is

$$|0\rangle \longmapsto \frac{1}{\sqrt{\Lambda}}\left(\sqrt{\|\boldsymbol{\psi}_h(0)\|}\,|0\rangle + \sum_{m=1}^{M_q}\sqrt{|w_m|\,\|\boldsymbol{g}_h(s_m)\|}\,|m\rangle\right), \tag{3.69}$$

together with a state oracle

$$|m\rangle\,|0\rangle \longmapsto \begin{cases} |0\rangle\left|\widehat{\boldsymbol{\psi}}_0\right\rangle, & m=0, \\ |m\rangle\,|\widehat{\boldsymbol{g}}_m\rangle, & 1\le m\le M_q, \end{cases} \tag{3.70}$$

and a SELECT operation that applies $U_h(T)$ for $m=0$ and $U_h(T-s_m)$ for $m\ge 1$. The phase of w_m is included in the controlled SELECT operation. After unpreparing the quadrature register, the successful branch is proportional to

$$\frac{1}{\Lambda}\left(U_h(T)\boldsymbol{\psi}_h(0) + \sum_{m=1}^{M_q} w_m U_h(T-s_m)\boldsymbol{g}_h(s_m)\right). \tag{3.71}$$

This spells out why Λ controls the success probability: the raw success amplitude is $\|\boldsymbol{\psi}_{h,M}(T)\|/\Lambda$. Since the propagators are unitary, the triangle inequality gives $\|\boldsymbol{\psi}_{h,M}(T)\| \le \Lambda$ always, with equality exactly when all the terms align in phase. The LCU therefore pays for cancellation: if the propagated initial wave and the emitted source waves interfere destructively at time T, the success amplitude shrinks in proportion. The source state in (3.70) is normalized, while its magnitude is carried by the coefficient register. Without such coherent source-history loading, (3.66) is only a classical quadrature formula, not yet a quantum LCU. This is the same issue that appears in LCHS and other inhomogeneous linear-ODE algorithms: the homogeneous propagator may be efficient, but the time-dependent forcing must also be coherently accessible.

Several common PDE sources have this structure. If $f(\boldsymbol{x}, t)$ is given by an analytic formula, the value oracle can compute it from the space-time grid indices. If the forcing has a separated form $f(\boldsymbol{x}, t) = \sum_r a_r(t) f_r(\boldsymbol{x})$, then the source history can be loaded from a small superposition over the separation index and the quadrature index. Boundary sources, such as the Neumann example below, are even more localized: the spatial state may be supported only on boundary degrees of freedom, while the time dependence is carried by a scalar function.

Consequently, after amplitude amplification, a schematic coherent cost is

$$\widetilde{O}\left(\frac{\Lambda}{\|\boldsymbol{\psi}_{h,M}(T)\|}\left[C_{\text{source}} + \frac{T}{h} + \log\frac{1}{\epsilon}\right]\right), \tag{3.72}$$

plus the cost needed to make the quadrature error sufficiently small. The formula keeps visible the two distinct issues: Hamiltonian simulation of the homogeneous wave and coherent preparation of the source superposition.

3.3.1 A mixed Dirichlet–Neumann boundary example

Boundary conditions deserve some care. In elliptic and parabolic problems, a Neumann boundary condition represents a prescribed flux. In mechanical wave problems, the analogous condition is often a prescribed traction. For the one-dimensional wave equation

$$u_{tt} - (a u_x)_x = f, \qquad a = c^2, \tag{3.73}$$

we consider the mixed boundary conditions

$$u(0, t) = 0, \qquad a u_x(L_{\text{box}}, t) = g_R(t). \tag{3.74}$$

The case $g_R = 0$ is the homogeneous Neumann condition. If u is a transverse displacement of a string or a longitudinal displacement of a bar, then $a u_x$ is the boundary force, or traction, after nondimensionalization.

For simplicity take $a = c^2$ constant and use a vertex-centered grid

$$x_j = jh, \qquad j = 0, \dots, N_{\text{cell}}, \qquad h = \frac{L_{\text{box}}}{N_{\text{cell}}}.$$

The left boundary value is fixed, $U_0(t) = 0$, while the right boundary value $U_{N_{\text{cell}}}(t)$ is retained as an unknown. Thus in this local example there are N_{cell} unknowns,

$$\boldsymbol{u}_h(t) = (U_1(t), \dots, U_{N_{\text{cell}}}(t))^\top.$$

For the interior nodes $1 \le j \le N_{\text{cell}} - 1$, the standard centered second difference gives

$$\ddot{U}_j = \frac{a}{h^2}\left(U_{j+1} - 2U_j + U_{j-1}\right) + f_j(t). \tag{3.75}$$

At the right boundary, a common strategy to incorporate a Neumann condition is to introduce a ghost point $U_{N_{\text{cell}}+1}$ and use the centered approximation

$$a\frac{U_{N_{\text{cell}}+1} - U_{N_{\text{cell}}-1}}{2h} = g_R(t). \tag{3.76}$$

Hence

$$U_{N_{\text{cell}}+1} = U_{N_{\text{cell}}-1} + \frac{2h}{a}g_R(t).$$

Substituting this into the centered second difference at $j = N_{\text{cell}}$ gives

$$\begin{aligned}\ddot{U}_{N_{\text{cell}}} &= \frac{a}{h^2}\left(U_{N_{\text{cell}}+1} - 2U_{N_{\text{cell}}} + U_{N_{\text{cell}}-1}\right) + f_{N_{\text{cell}}}(t) \\ &= \frac{2a}{h^2}\left(U_{N_{\text{cell}}-1} - U_{N_{\text{cell}}}\right) + \frac{2}{h}g_R(t) + f_{N_{\text{cell}}}(t).\end{aligned} \tag{3.77}$$

Therefore the semidiscrete equation can be written as

$$\ddot{\boldsymbol{u}}_h(t) + \widetilde{A}_h \boldsymbol{u}_h(t) = \boldsymbol{f}_h(t) + \frac{2g_R(t)}{h}\boldsymbol{e}_{N_{\text{cell}}}, \tag{3.78}$$

where the matrix from assembling the finite difference formulas is given by,

$$\widetilde{A}_h = \frac{a}{h^2}\begin{bmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -2 & 2 \end{bmatrix}. \tag{3.79}$$

The last row is different because the right boundary node represents a half cell. Consequently $\widetilde{A}_h$ is not symmetric in the standard Euclidean inner product.

This nonsymmetry is not a physical loss of self-adjointness. It can be attributed to a mass-weighting issue. Let

$$M_{DN} = \operatorname{diag}\left(1, 1, \ldots, 1, \frac{1}{2}\right), \tag{3.80}$$

where the harmless common factor h has been omitted, and define the symmetric stiffness matrix

$$K_{DN} = \frac{a}{h^2}\begin{bmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 1 \end{bmatrix}. \tag{3.81}$$

Then

$$\widetilde{A}_h = M_{DN}^{-1} K_{DN}. \tag{3.82}$$

This is exactly the mass-lumped finite element or finite-volume form of the mixed Dirichlet–Neumann discretization.

To obtain a symmetric matrix, introduce the mass-normalized variable

$$\boldsymbol{y}_h = M_{DN}^{1/2}\boldsymbol{u}_h.$$

Then (3.78) becomes

$$\ddot{\boldsymbol{y}}_h(t) + A_h^{DN}\boldsymbol{y}_h(t) = M_{DN}^{1/2}\left(\boldsymbol{f}_h(t) + \frac{2g_R(t)}{h}\boldsymbol{e}_{N_{\text{cell}}}\right), \tag{3.83}$$

where

$$A_h^{DN} = M_{DN}^{1/2}\widetilde{A}_h M_{DN}^{-1/2} = M_{DN}^{-1/2}K_{DN}M_{DN}^{-1/2} = (A_h^{DN})^\dagger. \tag{3.84}$$

Thus the apparently nonsymmetric matrix $\widetilde{A}_h$ is similar to a symmetric positive semidefinite matrix. For the homogeneous Neumann condition $g_R = 0$, the boundary contributes no forcing term. For nonzero traction, the boundary data appears as a time-dependent forcing term supported at the right boundary node.

This example illustrates a useful principle. Boundary conditions should be incorporated at the level of the semidiscrete variational, finite-volume, or finite-difference system before choosing the quantum encoding. A matrix that looks non-Hermitian in unweighted coordinates may become Hermitian after the proper mass normalization. Genuine non-Hermitian dilation is needed only when the discretization itself contains dissipation, absorbing layers, upwinding, or other non-self-adjoint mechanisms.

The extension of this simple one-dimensional ghost-point calculation to higher-dimensional domains is more delicate. On tensor-product grids, one can derive analogous boundary rows face by face. On curved or unstructured domains, Neumann and traction conditions are most naturally handled through the weak form and the boundary load vector. Developing efficient block-encodings that preserve this mass-weighted self-adjoint structure for general high-dimensional geometries remains an interesting open problem.

3.4 QFT diagonalization, fast forwarding, and splitting

Constant-coefficient finite differences on uniform tensor-product grids have more structure than a generic sparse Hamiltonian: the spatial operator is diagonalized by Fourier, sine, or cosine transforms. Recent QFT-based wave-equation circuits exploit exactly this structure [86, 68].

The analogy with the classical FFT solver is close but not identical. Classically, one transforms all grid values to Fourier space, evolves each mode, and transforms back. Quantumly, the Fourier transform is applied coherently to the amplitude-encoded state. The mode index is then available in a register, so the mode frequency can be computed by reversible arithmetic and used to control a rotation. The circuit acts on all Fourier modes in superposition. This is why the cost is polylogarithmic in the number of grid points in the ideal structured setting, rather than proportional to the number of modes.

3.4.1 Uniform grids with periodic boundary conditions

Consider the periodic one-dimensional wave equation with $N = 2^n$ grid points. This N denotes the periodic register size, not the number of interior Dirichlet unknowns used earlier. If a nonperiodic discretization has a non-power-of-two number of degrees of freedom, one pads the register as in Chapter 2. The discrete Fourier transform diagonalizes the positive Laplacian:

$$F_N A_h F_N^\dagger = \mathrm{diag}(\omega_k^2), \qquad \omega_k = \frac{2c}{h}\left|\sin\left(\frac{\pi k}{N}\right)\right|, \qquad k = 0, \ldots, N-1. \tag{3.85}$$

After the transformation, each Fourier mode satisfies a decoupled oscillator equation

$$\ddot{\widehat{u}}_k(t) + \omega_k^2 \widehat{u}_k(t) = 0. \tag{3.86}$$

For $\omega_k > 0$, define the two-component mode energy vector

$$\boldsymbol{\chi}_k(t) = \begin{bmatrix} \omega_k \widehat{u}_k(t) \\ \widehat{v}_k(t) \end{bmatrix}, \qquad \widehat{v}_k = \dot{\widehat{u}}_k. \tag{3.87}$$

It obeys a 2 by 2 Schrödinger equation,

$$\frac{d}{dt}\boldsymbol{\chi}_k(t) = \omega_k \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \boldsymbol{\chi}_k(t) = i\omega_k Y \boldsymbol{\chi}_k(t), \tag{3.88}$$

where Y is the Pauli matrix from Chapter 1. Hence

$$\boldsymbol{\chi}_k(T) = e^{i\omega_k T Y} \boldsymbol{\chi}_k(0), \tag{3.89}$$

or, equivalently, the two eigenstates of Y acquire phases $e^{\pm i\omega_k T}$.

Although there are N Fourier modes, the quantum circuit does not apply N rotations sequentially. The mode index k is stored in a binary register. Reversible arithmetic computes $\omega_k T$ coherently in superposition, and a single controlled rotation acts on an energy qubit with angle determined by that register. The workflow is

(i) apply the QFT to the spatial register;

(ii) reversibly compute $\omega_k T$ modulo 2π from the binary index k;

(iii) apply the controlled Y-rotation $e^{i\omega_k T Y}$;

(iv) uncompute the angle register and apply the inverse QFT.

This is one coherent circuit acting on the Fourier register, not a loop over modes. Classically, applying all N mode rotations would require at least $O(N)$ arithmetic operations after an FFT. Quantumly, the register contains a superposition of all k, and the same reversible arithmetic circuit applies the rotation associated with the binary value of k to every amplitude in superposition. The price is that the result is still a quantum state: one cannot read all rotated Fourier coefficients without $O(N)$ measurements.

The exact n-qubit QFT costs $O(n^2) = O(\log^2 N)$ elementary gates, as reviewed in Section 1.7. The additional phase arithmetic has cost polynomial in n and in the number of precision bits.

Thus the explicit spectral circuit does not incur the generic multiplicative T/h Hamiltonian-simulation count. The factor $1/h$ appears in the magnitude of the phase $\omega_k T$, and therefore in the arithmetic precision needed to compute that phase, rather than as the number of simulated time steps.

This is the wave-equation analogue of the Fourier-space reciprocal-filter construction used for Poisson-type equations in [36]. The elliptic filter $\lambda_{\boldsymbol{k}}^{-1}$ is replaced by the unitary phase or rotation $e^{\pm i\omega_{\boldsymbol{k}} T}$. In several dimensions,

$$\lambda_{\boldsymbol{k}} = \frac{4}{h^2}\sum_{\ell=1}^{d} \sin^2\left(\frac{\pi k_\ell}{N_\ell}\right), \qquad \omega_{\boldsymbol{k}} = c\sqrt{\lambda_{\boldsymbol{k}}}, \tag{3.90}$$

so the arithmetic includes the summation and square-root step. The advantage is that the structured spectral formula can bypass generic sparse Hamiltonian simulation. The limitation is equally important: this fast-forwarding relies on separability and an explicit diagonal symbol. It does not extend directly to general finite element meshes or variable-coefficient wave speeds.

For Dirichlet and Neumann conditions on rectangular grids, the QFT is replaced by quantum sine and cosine transforms. These transforms are QFT relatives obtained by even or odd extensions and have comparable polylogarithmic circuit cost; see, for example, the QFT discussion in [71, 23]. In d dimensions, tensor-product transforms give formulas analogous to (3.90), with the appropriate boundary-dependent modification. The QFT-based circuits of Wright et al. and Lubasch et al. also exploit smooth initial data to reduce circuit demands, but state preparation and observable readout remain nontrivial parts of the end-to-end cost [86, 68].

3.4.2 Klein–Gordon type equations

A useful extension is the Klein–Gordon equation with a spatially varying mass term,

$$u_{tt} - c^2\Delta u + m(\boldsymbol{x})^2 u = 0. \tag{3.91}$$

The term $m(\boldsymbol{x})^2 u$ is a local restoring force. When m is constant, it creates a spectral gap and makes low-frequency waves oscillate even at zero spatial frequency. When $m(\boldsymbol{x})$ varies, it acts like a spatially varying local oscillator strength, producing scattering, localization, or trapping effects in much the same way that a potential modifies Schrödinger dynamics.

After discretization and mass normalization, write

$$\ddot{\boldsymbol{y}}_h + (A_{K,h} + A_{M,h})\boldsymbol{y}_h = \boldsymbol{0}, \tag{3.92}$$

where $A_{K,h}$ is the discretized $-\Delta$ operator and $A_{M,h}$ is the positive diagonal matrix associated with $m(\boldsymbol{x})^2$. Choose

$$A_{K,h} = G_{K,h}^{\dagger} G_{K,h}, \qquad A_{M,h} = G_{M,h}^{\dagger} G_{M,h}, \qquad G_{M,h} = A_{M,h}^{1/2}, \tag{3.93}$$

where $G_{M,h}$ is diagonal in physical space. Stack the factors,

$$G_h = \begin{bmatrix} G_{K,h} \\ G_{M,h} \end{bmatrix}, \qquad G_h^{\dagger} G_h = A_{K,h} + A_{M,h}. \tag{3.94}$$

The stacked form is more than notation. It separates the part of the energy stored in spatial deformation from the part stored in the local oscillator term. When $m(\boldsymbol{x})$ is diagonal in physical space, $G_{M,h}$ is also diagonal after a simple finite difference discretization and is therefore easy to block encode or implement by controlled rotations. The kinetic factor $G_{K,h}$, on the other hand, is diagonal in Fourier space for constant coefficients. This is the reason a split-step method is natural.

The energy state has two pressure/strain blocks sharing one velocity block,

$$\boldsymbol{\psi}_h = \begin{bmatrix} \boldsymbol{p}_{K,h} \\ \boldsymbol{p}_{M,h} \\ \boldsymbol{v}_h \end{bmatrix}, \qquad \boldsymbol{p}_{K,h} = G_{K,h}\boldsymbol{y}_h, \quad \boldsymbol{p}_{M,h} = G_{M,h}\boldsymbol{y}_h, \quad \boldsymbol{v}_h = \dot{\boldsymbol{y}}_h, \tag{3.95}$$

and again $\boldsymbol{\psi}_h$ solves a Schrödinger equation. We notice that its Hamiltonian splits naturally as $H_h = H_{K,h} + H_{M,h}$, with

$$H_{K,h} = i \begin{bmatrix} 0 & 0 & G_{K,h} \\ 0 & 0 & 0 \\ -G_{K,h}^\dagger & 0 & 0 \end{bmatrix}, \qquad H_{M,h} = i \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & G_{M,h} \\ 0 & -G_{M,h}^\dagger & 0 \end{bmatrix}. \tag{3.96}$$

Both terms are Hermitian. The $H_{M,h}$ evolution consists of independent physical-space rotations, while $H_{K,h}$ becomes a collection of Fourier-space rotations after the QFT.

A direct approach for simulating $\boldsymbol{\psi}_h(t)$ is Hamiltonian simulations by qubitization of QSVT applied to H_h. On the other hand, this splitting in equation (3.96) is the wave-equation analogue of split-operator methods for the Schrödinger equation. The mass term is diagonal in physical space, so it is cheap when $m(\boldsymbol{x})$ can be evaluated from the grid index. The Laplacian is diagonal in Fourier space, so it is cheap on a uniform periodic or separable grid. The noncommutativity of these two diagonal representations is exactly what the splitting error measures.

The most widely used splitting scheme in classical algorithms is Strang splitting. For time step of size Δt, it approximates the evolution by,

$$e^{-i\Delta t H_h} \approx e^{-i\Delta t H_{M,h}/2} e^{-i\Delta t H_{K,h}} e^{-i\Delta t H_{M,h}/2}. \tag{3.97}$$

If the relevant nested commutators are bounded, the one-step error is

$$\|e^{-i\Delta t(H_{K,h}+H_{M,h})} - S_{\Delta t}\| \le C_{\mathrm{comm}} \Delta t^3, \tag{3.98}$$

and the global error over time T is $O(C_{\mathrm{comm}} T \Delta t^2)$. This constant is generally mesh dependent. If $G_{K,h}$ has size $O(h^{-1})$ and $G_{M,h}$ is bounded independently of h, then a rough estimate for the nested commutators gives the scaling

$$C_{\mathrm{comm}} = O(h^{-2}). \tag{3.99}$$

Thus a Strang calculation with error $O(\epsilon)$ over time T may require roughly

$$T/\Delta t = O\Big(T^{3/2} h^{-1} \epsilon^{-1/2}\Big)$$

steps in a worst-case norm estimate. Compared with the generic count $\widetilde{O}(T/h)$ of qubitization, the number of steps is larger by a factor $T^{1/2}\epsilon^{-1/2}$; what splitting buys in exchange is the

structure of each step. The method can still be attractive because each step uses very structured QFT and diagonal-rotation circuits. It should not, however, be read as a mesh-independent splitting method. Each step follows the workflow

$$\begin{aligned}&\text{physical-space rotation} \longrightarrow \text{QFT} \longrightarrow \text{Fourier-space rotation}\\ &\longrightarrow \text{QFT}^\dagger \longrightarrow \text{physical-space rotation.}\end{aligned} \tag{3.100}$$

Higher-order Trotter splitting schemes can also be applied to reduce the dependence on T and ϵ. The commutator error scaling has been extensively studied [20].

3.5 General meshes: access, state preparation, and readout

For unstructured finite element meshes, variable coefficients, complicated domains, and nonseparable boundary conditions, Fourier diagonalization is unavailable. The robust approach is to block encode the sparse factor G_h and simulate the Hermitian Hamiltonian (3.42). Its generic cost has already been stated in (3.57). Figure 3.1 summarizes the resulting pipeline; the rest of this section walks through its three stages—input, evolution, and readout—and records what each stage costs.

This is the point at which the access model becomes part of the numerical method. In a finite difference code, one might think of G_h as a stencil. In a finite element code, one may instead access it through local element matrices, quadrature rules, and incidence relations between elements and degrees of freedom. Both descriptions are sparse, but they lead to different reversible oracles. The book-level abstraction is a block encoding of G_h/α_G; the implementation-level question is how the mesh data and coefficient data are made coherent.

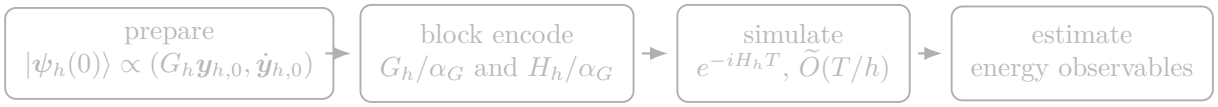

Figure 3.1: The generic wave-equation pipeline. The homogeneous energy-variable evolution is unitary and requires no final postselection. The Duhamel–LCU construction for sources introduces a separate source-loading and success-amplitude factor.

The input state is proportional to

$$\boldsymbol{\psi}_h(0) = \begin{bmatrix} G_h \boldsymbol{y}_h(0) \\ \dot{\boldsymbol{y}}_h(0) \end{bmatrix}. \tag{3.101}$$

Thus state preparation requires access not only to the initial displacement and velocity, but also to the discrete gradient of the displacement. On structured grids this can be generated by arithmetic circuits or by applying a block encoding of G_h. For mass-lumped finite elements, one first prepares the coefficient vectors, applies the diagonal mass normalization, and then applies the local gradient factor.

For smooth initial data on a tensor-product grid, Grover–Rudolph type state preparation or Fourier preparation may be appropriate. For finite element data, a more realistic input may be a mesh-based data structure or a classical preprocessing routine that prepares nodal values. In

either case, the cost of preparing $G_h \boldsymbol{y}_{h,0}$ should not be silently hidden: it is part of the end-to-end algorithm, just as assembling a stiffness matrix or applying a gradient operator is part of a classical workflow.

The output quantum state is the normalized pressure–velocity state

$$|\boldsymbol{\psi}_h(T)\rangle := \frac{\boldsymbol{\psi}_h(T)}{\|\boldsymbol{\psi}_h(T)\|}, \qquad \boldsymbol{\psi}_h(T) = \begin{bmatrix} G_h \boldsymbol{y}_h(T) \\ \dot{\boldsymbol{y}}_h(T) \end{bmatrix}. \tag{3.102}$$

For an observable $O = O^\dagger$ acting on the energy variables, the normalized quantity of interest is

$$\mu_O(T) = \langle \boldsymbol{\psi}_h(T)| \, O \, |\boldsymbol{\psi}_h(T)\rangle = \frac{\boldsymbol{\psi}_h(T)^\dagger O \boldsymbol{\psi}_h(T)}{\|\boldsymbol{\psi}_h(T)\|^2}. \tag{3.103}$$

This is the observable-estimation problem introduced in Section 1.4. After rescaling O, we may assume $\|O\| \le 1$. One preparation of $|\boldsymbol{\psi}_h(T)\rangle$ costs

$$Q_{\mathrm{sim}}(T, \epsilon) = \widetilde{O}\left(\frac{T}{h} + \log\frac{1}{\epsilon}\right) \tag{3.104}$$

queries to the block encoding of the wave Hamiltonian.

With direct sampling, $O(\epsilon^{-2}\log(1/\delta))$ independent preparations and measurements estimate $\mu_O(T)$ to additive error ϵ with failure probability at most δ. Hence

$$Q_{\mathrm{tot}}^{\mathrm{samp}} = \widetilde{O}\left(\left(\frac{T}{h} + \log\frac{1}{\epsilon}\right)\frac{1}{\epsilon^2}\log\frac{1}{\delta}\right). \tag{3.105}$$

With amplitude estimation, the measurement dependence improves to $O(\epsilon^{-1})$, provided one can coherently apply the state-preparation circuit and its inverse:

$$Q_{\mathrm{tot}}^{\mathrm{AE}} = \widetilde{O}\left(\left(\frac{T}{h} + \log\frac{1}{\epsilon}\right)\frac{1}{\epsilon}\log\frac{1}{\delta}\right). \tag{3.106}$$

This is the same sampling-versus-amplitude-estimation distinction discussed in Chapter 1. The improvement in measurement complexity comes at the price of greater coherent depth. For near-term or early fault-tolerant devices, direct sampling may be more realistic; for asymptotic resource estimates, amplitude estimation is the natural benchmark. In either case, the measurement cost multiplies the cost of preparing the wave state, so it should not be omitted from an end-to-end PDE complexity estimate.

Finally, the normalized expectation can be converted into the corresponding unnormalized physical quadratic form:

$$\boldsymbol{\psi}_h(T)^\dagger O \boldsymbol{\psi}_h(T) = \|\boldsymbol{\psi}_h(T)\|^2 \mu_O(T). \tag{3.107}$$

For the homogeneous wave equation,

$$\|\boldsymbol{\psi}_h(T)\| = \|\boldsymbol{\psi}_h(0)\|, \tag{3.108}$$

so this normalization factor is determined by the initial discrete energy. This is simpler than the nonunitary linear-system and heat-equation settings discussed in Subsection 1.10.3, where the output norm may itself be an unknown quantity.

Recovering every displacement coefficient is not generally an efficient quantum output task. The favorable setting is instead one in which the desired result is a small collection of energy fractions, local energies, modal weights, fluxes, or other observables. If displacement observables are essential, one can carry the displacement as an additional dynamical block, as described next.

3.5.1 An augmented displacement formulation

So far, we have relied on the first order form, using the pressure and velocity as the primary variables, ot leverage efficient Hamiltonian simulations. To include the displacement in the simulation, we can define

$$\boldsymbol{z}_h(t) = \begin{bmatrix} \boldsymbol{y}_h(t) \\ \boldsymbol{p}_h(t) \\ \boldsymbol{v}_h(t) \end{bmatrix}, \qquad \boldsymbol{p}_h = G_h \boldsymbol{y}_h, \quad \boldsymbol{v}_h = \dot{\boldsymbol{y}}_h. \tag{3.109}$$

Then the time evolution becomes

$$\frac{d}{dt}\boldsymbol{z}_h(t) = L_{\text{aug}}\boldsymbol{z}_h(t), \qquad L_{\text{aug}} = \begin{bmatrix} 0 & 0 & I \\ 0 & 0 & G_h \\ 0 & -G_h^\dagger & 0 \end{bmatrix}. \tag{3.110}$$

The accumulator equation $\dot{\boldsymbol{y}}_h = \boldsymbol{v}_h$ makes L_{aug} non-skew-Hermitian in the Euclidean inner product. It therefore cannot be simulated directly by ordinary Hamiltonian simulation. The dilation introduced in the next section gives a systematic way to map this augmented dynamics, as well as dissipative upwind dynamics, to a Schrödinger equation on a larger space. In particular, since the non-Hermitian part of L_{aug} only scales $O(1)$, the overall time complexity is still $O(T/h)$.

The need for this augmented system is purely an output issue. The homogeneous pressure–velocity dynamics already gives a unitary quantum evolution, but it does not store $\boldsymbol{y}_h$ as a state component. Carrying $\boldsymbol{y}_h$ turns the evolution into an accumulator system, analogous to integrating velocity to recover displacement in a classical code. Accumulators are useful, but they are not energy-preserving by themselves, which is why dilation rather than direct Hamiltonian simulation is needed.

3.6 First-order hyperbolic systems, Schrödingerization, and dilation

Many models in transport, acoustics, elasticity, and electromagnetism are written directly as first-order systems. These systems are closer to the pressure–velocity formulation than to the second-order displacement formulation. The main new issue is that stable finite difference methods for first-order hyperbolic equations often create numerical dissipation. That dissipation is essential classically, but it destroys unitarity and therefore forces us to use some form of embedding or dilation. This is the far end of the conservation dial: the departure from unitarity is no longer a bookkeeping matter, as it was for sources, but is built into the discretization itself. The three frameworks presented below—Schrödingerization, LCHS, and moment-matching dilation—are three routes to the same destination, a unitary evolution on a larger space whose projection reproduces the stable nonunitary scheme.

Consider the constant-coefficient problem

$$\boldsymbol{u}_t + A\boldsymbol{u}_x = \boldsymbol{0}, \qquad \boldsymbol{u}(x,t) \in \mathbb{R}^m. \tag{3.111}$$

The PDE system is called hyperbolic if A is diagonalizable with real eigenvalues,

$$A = S\Lambda S^{-1}, \qquad \Lambda = \text{diag}(\lambda_1, \ldots, \lambda_m). \tag{3.112}$$

The characteristic variables $\boldsymbol{w} = S^{-1}\boldsymbol{u}$ satisfy decoupled advection equations

$$(w_j)_t + \lambda_j (w_j)_x = 0. \tag{3.113}$$

This is the classical reason that upwinding is built from the eigendecomposition of the flux matrix. Symmetrizable systems can be treated similarly after changing the inner product.

Define

$$\Lambda^+ = \mathrm{diag}(\max(\lambda_j, 0)), \qquad \Lambda^- = \mathrm{diag}(\min(\lambda_j, 0)), \tag{3.114}$$

and

$$A^+ = S\Lambda^+ S^{-1}, \qquad A^- = S\Lambda^- S^{-1}, \qquad |A| = A^+ - A^- = S|\Lambda|S^{-1}. \tag{3.115}$$

Let $\boldsymbol{u}_j(t)$ denote the grid value at x_j. The semidiscrete upwind method can be expressed as the matrix-vector level,

$$\dot{\boldsymbol{u}}_j + A^+ \frac{\boldsymbol{u}_j - \boldsymbol{u}_{j-1}}{h} + A^- \frac{\boldsymbol{u}_{j+1} - \boldsymbol{u}_j}{h} = \boldsymbol{f}_j(t), \tag{3.116}$$

where $\boldsymbol{f}_j(t)$ contains body forcing or boundary contributions. Equivalently,

$$\dot{\boldsymbol{u}}_j = -A\frac{\boldsymbol{u}_{j+1} - \boldsymbol{u}_{j-1}}{2h} + \frac{h}{2}|A|\frac{\boldsymbol{u}_{j+1} - 2\boldsymbol{u}_j + \boldsymbol{u}_{j-1}}{h^2} + \boldsymbol{f}_j(t). \tag{3.117}$$

The first term is the centered transport part. The second term is numerical viscosity. This is the semidiscrete form of the familiar finite-difference principle: centered differencing is often nondissipative, while upwind differencing adds just enough damping to control one-sided propagation.

For the global vector

$$\boldsymbol{u}_h(t) = (\boldsymbol{u}_0(t)^T, \ldots, \boldsymbol{u}_{N-1}(t)^T)^T, \tag{3.118}$$

the semidiscrete equation has the form

$$\dot{\boldsymbol{u}}_h(t) = L_h^{\mathrm{up}} \boldsymbol{u}_h(t) + \boldsymbol{f}_h(t). \tag{3.119}$$

This is a useful example of the tension between numerical stability and quantum unitarity. A centered discretization of a symmetric hyperbolic system is naturally skew-Hermitian, but it may be oscillatory near discontinuities or sharp gradients. Upwinding repairs the classical stability problem by adding dissipation. The goal is therefore not to discard upwinding in order to keep a unitary equation, but to embed the stable nonunitary scheme into a larger unitary dynamics.

3.6.1 Schrödingerization: the PDE-to-Schrödinger transformation

A systematic way to embed nonunitary PDE dynamics into Schrödinger-type dynamics was introduced by Jin and collaborators under the name *Schrödingerization* [47, 43]. The basic idea is to add one auxiliary continuous variable so that the dissipative part of the PDE becomes a derivative in that new variable. The resulting equation is a Schrödinger equation on an enlarged space.

For the semidiscrete flow

$$\dot{\boldsymbol{x}}(t) = (-iH(t) + K(t))\boldsymbol{x}(t), \qquad H(t) = H(t)^\dagger, \qquad K(t) = K(t)^\dagger, \tag{3.120}$$

introduce an auxiliary coordinate $p \geq 0$ and set, formally,

$$\boldsymbol{\Psi}(t,p) = e^{-p}\boldsymbol{x}(t). \tag{3.121}$$

Since $\partial_p \boldsymbol{\Psi} = -\boldsymbol{\Psi}$, the term $K(t)\boldsymbol{x}(t)$ can be represented as $-K(t)\partial_p \boldsymbol{\Psi}(t,p)$. Thus (3.120) is lifted to

$$i\partial_t \boldsymbol{\Psi}(t,p) = [H(t) - iK(t)\partial_p]\,\boldsymbol{\Psi}(t,p). \tag{3.122}$$

The operator $-i\partial_p$ is Hermitian after choosing an appropriate domain or Fourier representation in the auxiliary variable, and therefore the right-hand side of (3.122) is a Hamiltonian on the enlarged space. Evaluation at $p = 0$ (or a different point) recovers the original solution:

$$\boldsymbol{x}(t) = \boldsymbol{\Psi}(t,0).$$

This formal calculation captures the essential mechanism. In rigorous algorithms one must choose an auxiliary interval or Fourier representation, discretize the p-variable, control boundary errors, and account for the success probability of evaluating or postselecting the auxiliary state. These are not minor implementation details, but the conceptual point is powerful: dissipation in the original PDE is converted into transport or phase evolution in an extra coordinate. In this sense, Schrödingerization is one of the first systematic PDE-to-Schrödinger frameworks for quantum simulation, complementary to the finite-dimensional block-encoding approach used throughout this book.

3.6.2 LCHS: an LCU of Hamiltonian simulations

Another way to view the same embedding problem is through the LCHS method of An, Liu, and Lin [4], already encountered for sources in Section 3.3. LCHS seeks a representation of a nonunitary flow as a linear combination, or integral, of unitary Hamiltonian simulations. This makes the connection to the LCU primitive of Chapter 1 particularly explicit.

For an autonomous generator

$$L = -iH + K, \qquad H = H^\dagger, \qquad K = K^\dagger,$$

one aims for an identity or approximation of the schematic form

$$e^{t(-iH+K)} \approx \sum_{m=1}^{M} c_m\, e^{-it(H+\eta_m K)}. \tag{3.123}$$

Each term is a Hamiltonian simulation with Hamiltonian $H + \eta_m K$. Once the coefficients c_m and parameters η_m are chosen, the algorithm has the standard LCU structure:

$$\text{PREPARE} : |0\rangle \mapsto \frac{1}{\sqrt{\Lambda}} \sum_m \sqrt{|c_m|}\,|m\rangle\,, \qquad \Lambda = \sum_m |c_m|,$$

followed by

$$\text{SELECT} : |m\rangle\,|\psi\rangle \mapsto |m\rangle\; e^{-it(H+\eta_m K)}\,|\psi\rangle\,,$$

and postselection on the coefficient register. Thus the LCU normalization Λ, the longest simulation time, and the cost of block encoding $H + \eta_m K$ determine the complexity.

In continuous form, (3.123) becomes

$$e^{t(-iH+K)} \approx \int_{\mathbb{R}} w(\eta)\, e^{-it(H+\eta K)}\, d\eta. \tag{3.124}$$

The LCHS formula needs to assume that K is negative definite, or equivalently, the left hand side induces a contraction. After quadrature, this is exactly a linear combination of Hamiltonian simulations. This representation is especially appealing in numerical PDEs: one keeps the stable nonunitary discretization, but simulates a family of Hermitian Hamiltonians instead of the non-Hermitian matrix directly.

The LCHS viewpoint is closely related to Schrödingerization. In both cases, a dissipative or nonunitary term is represented by a superposition of unitary evolutions in an enlarged space. The difference is mostly organizational: Schrödingerization emphasizes the auxiliary PDE and continuous-variable Hamiltonian, whereas LCHS emphasizes the quadrature and LCU implementation.

3.6.3 Moment-matching dilation

The moment-matching dilation framework of [58] abstracts both of these ideas. Consider again

$$\dot{\boldsymbol{x}}(t) = L(t)\boldsymbol{x}(t), \qquad L(t) = -iH(t) + K(t), \qquad H(t) = H(t)^\dagger, \qquad K(t) = K(t)^\dagger. \tag{3.125}$$

The natural question is: what ancillary operator F, right vector $|r\rangle$, and evaluation functional $\langle l|$ make a projected unitary evolution reproduce this nonunitary flow exactly?

Let $F^\dagger = -F$ act on an ancillary space, and define the dilated Hamiltonian

$$\widetilde{H}(t) = I_A \otimes H(t) + iF \otimes K(t). \tag{3.126}$$

It is Hermitian because iF and $K(t)$ are Hermitian. Expanding its propagator in a Taylor series in the autonomous case, or a Dyson series in the time-dependent case, shows that every term containing q occurrences of K carries the ancillary coefficient

$$m_q := \langle l|F^q|r\rangle. \tag{3.127}$$

A detail is worth making explicit, because it is what makes the construction work. Within a given expansion term, the H factors and the K factors are interleaved in some time order; but the H factors act as the identity on the ancillary space, so the ancillary operator product is F^q for *every* interleaving. The ancillary coefficient therefore depends only on the count q, not on the positions of the K factors, and a single sequence of scalar moments controls the entire series. Thus exact equality is obtained if

$$\langle l|F^q|r\rangle = 1, \qquad q = 0, 1, 2, \ldots. \tag{3.128}$$

Under these identities,

$$(\langle l| \otimes I)\mathcal{T}\exp\left[-i\int_0^t \widetilde{H}(s)\, ds\right](|r\rangle \otimes I) = \mathcal{T}\exp\left[\int_0^t L(s)\, ds\right]. \tag{3.129}$$

Thus matching every moment gives an exact dilation, while matching only the first several moments gives a finite-order approximation. In exact function-space constructions, the encoding

vector or evaluation functional may live in a larger dual space rather than being a normalizable quantum state; a numerical algorithm approximates them by truncation and discretization.

Schrödingerization is recovered by taking

$$F = -\partial_p, \qquad r(p) = e^{-p}, \qquad \langle l|\varphi\rangle = \varphi(0). \tag{3.130}$$

Since $F^q r = r$ for every $q \geq 0$, the moment identities hold; verifying this is the one-line computation requested in the exercises. Other choices of the moment triple produce differential, integral, pseudodifferential, finite-difference, and Bargmann–Fock dilations [58].

The analogy with reduced-order modeling is intentional. In rational Krylov and moment-matching methods, one chooses a small subspace so that selected transfer-function moments are reproduced. Here one chooses an ancillary dynamics so that the moments of the dissipative part $K(t)$ are reproduced by projected unitary evolution. The reward is that a nonunitary numerical flow can be represented by a unitary flow on a larger space; the price is ancillary dimension, projection success probability, and discretization of the ancillary variable.

Finally, if iF has spectral resolution

$$iF = \int_{\mathbb{R}} \eta \, dE_F(\eta), \tag{3.131}$$

then each ancillary spectral sector evolves under $H + \eta K$. Projection onto $\langle l|$ and $|r\rangle$ gives an integral representation

$$e^{t(-iH+K)} = \int_{\mathbb{R}} w_{l,r}(\eta) e^{-it(H+\eta K)} \, d\eta, \tag{3.132}$$

at least formally in the autonomous case. After quadrature, this becomes the LCHS/LCU form in (3.123). Thus Schrödingerization, LCHS, and moment-matching dilation are three views of the same underlying principle: represent a stable nonunitary flow as the projection of a unitary flow in a larger space. The Gaussian-transmutation representation of the heat semigroup in Chapter 4 is built in exactly this spirit: a dissipative flow written as an integral of unitary evolutions, with an explicit weight and an explicit quadrature.

3.6.4 Implication for upwind hyperbolic discretizations

We now return to the upwind system

$$\dot{\boldsymbol{u}}_h = L_h^{\mathrm{up}} \boldsymbol{u}_h + \boldsymbol{f}_h(t), \qquad L_h^{\mathrm{up}} = -iH_h + K_h.$$

When A is Hermitian, the centered transport part gives H_h, while the upwind viscosity gives $K_h \preceq 0$. The preceding discussion says that we do not need to abandon this stable discretization in order to make a quantum algorithm. Instead, one can choose a dilation or LCHS representation and simulate Hermitian Hamiltonians of the form

$$H_h + \eta K_h.$$

For fixed spatial dimension and fixed system size m, the operator norms scale as

$$\|H_h\| + \|K_h\| = O\left(\frac{\max_j |\lambda_j|}{h}\right).$$

Therefore the coherent simulation scale for the homogeneous upwind evolution is expected to be

$$\widetilde{O}\left(\frac{T \max_j |\lambda_j|}{h}\right), \tag{3.133}$$

up to logarithmic factors, dilation-discretization costs, and postselection or LCU normalization factors. This is the same CFL-type scale that appears in classical explicit hyperbolic solvers, but without the factor corresponding to updating all grid values. As throughout this chapter, the quantum advantage can only be meaningful for state preparation, sampling, or low-dimensional observables, not for printing the entire solution vector.

For nonzero forcing or boundary terms, one combines the dilation of the homogeneous propagator with the Duhamel and LCU construction described in Section 3.3. The source history must be loaded coherently, and its norms enter the LCU normalization. Thus the final workflow is:

1. construct a stable upwind or flux-splitting semidiscretization;
2. write the global ODE as $\dot{\boldsymbol{u}}_h = L_h^{\mathrm{up}} \boldsymbol{u}_h + \boldsymbol{f}_h(t)$;
3. separate $L_h^{\mathrm{up}} = -iH_h + K_h$;
4. block encode H_h and K_h;
5. choose a Schrödingerization, LCHS, or moment-matching dilation;
6. simulate the enlarged Hermitian dynamics and estimate the desired observable.

The same dilation principle also applies to the augmented displacement generator in (3.110). In that case the nonunitarity is introduced artificially to carry an additional displacement block; in the upwind case, it is introduced by the stable numerical discretization itself.

3.7 Summary and outlook

The main lesson of this chapter is that hyperbolic PDEs enter quantum algorithms through their energy structure. Finite differences and finite elements give

$$M_h \ddot{\boldsymbol{q}}_h + K_h \boldsymbol{q}_h = \boldsymbol{b}_h(t). \tag{3.134}$$

After mass normalization and factorization $A_h = M_h^{-1/2} K_h M_h^{-1/2} = G_h^\dagger G_h$, the homogeneous equation becomes

$$i\dot{\boldsymbol{\psi}}_h = H_h \boldsymbol{\psi}_h, \qquad H_h = i \begin{bmatrix} 0 & G_h \\ -G_h^\dagger & 0 \end{bmatrix}. \tag{3.135}$$

For general meshes, optimal Hamiltonian simulation has the leading scale $\widetilde{O}(T/h)$. Explicit Fourier diagonalization can do better for special constant-coefficient tensor-product problems. Upwind discretizations introduce controlled nonunitarity and motivate moment-matching dilation.

From a numerical-analysis perspective, the message is that the choice of variables matters as much as the choice of discretization. The same second-order wave equation can lead to

a displacement–velocity system, a pressure–velocity Hamiltonian, an augmented accumulator system, or a dissipative upwind system. These formulations are classically equivalent or closely related, but they lead to different quantum primitives: Hamiltonian simulation, QFT phase arithmetic, LCU for sources, or dilation for nonunitary dynamics.

Boundary conditions. Nonhomogeneous Dirichlet, Neumann, and Robin data can be converted into source terms and treated by the Duhamel–LCU construction. Absorbing layers, impedance conditions, and perfectly matched layers generally introduce dissipation and are more naturally handled by dilation. From a numerical viewpoint, this is the familiar distinction between conservative boundary treatments and energy-absorbing ones. From a quantum viewpoint, it is the distinction between unitary Hamiltonian simulation and nonunitary evolution with a success probability or dilation overhead. Efficient preparation of liftings and boundary-load states remains a problem-specific issue.

Fast-forwardable wave dynamics. The QFT example shows that long-time wave propagation can be fast-forwarded when both the eigenbasis and dispersion relation are efficiently computable. This is a very special situation. The circuit computes the phase $\omega_{\boldsymbol{k}} T$ directly instead of simulating the Hamiltonian for time T. Characterizing other PDE discretizations with this structure, while respecting the generic no-fast-forwarding obstruction, is an important direction. Possible examples include separable media, periodic lattice models, and certain exactly diagonalizable discretizations.

Multiscale wave dynamics. High-contrast media, small geometric features, and locally refined meshes introduce several spatial scales. The finest scale still controls $\|G_h\|$ and hence the generic Hamiltonian-simulation normalization. This mirrors classical CFL restrictions, where the smallest cells and largest wave speeds dictate the time step. Multilevel, homogenization, or reduced-basis ideas may be needed to prevent the smallest mesh size from determining the full quantum cost.

Maxwell equations. Maxwell's equations are already first order and possess a natural energy inner product. Compatible curl discretizations, and edge elements can lead directly to Hermitian or skew-Hermitian block systems, while conductivity, absorbing boundaries, and material dispersion introduce nonunitary terms. They are therefore a natural next application of both Hamiltonian simulation and dilation. A careful treatment would also need to address gauge constraints, divergence constraints, and the preparation and measurement of physically meaningful field observables.

More general spatial discretizations. Discontinuous Galerkin, spectral-element, mixed, mimetic, and high-order finite difference methods may offer better accuracy per degree of freedom than the lowest-order schemes used here. Their quantum suitability depends not only on sparsity, but also on mass-matrix structure, factorization, block-encoding normalization, state preparation, and the observable to be measured. The most promising discretization for a quantum algorithm may therefore not be the one with the fewest classical floating-point operations. It may instead be the one whose algebraic structure leads to the cleanest block encoding and the most useful quantum output state.

What to remember

- The homogeneous wave equation becomes a Schrödinger equation after spatial discretization, mass normalization, and passage to pressure/strain and velocity variables.
- The generic Hamiltonian-simulation normalization is $\|G_h\| = \Theta(h^{-1})$, so the black-box wave cost scales like $\widetilde{O}(T/h)$, not T/h^2.
- QFT methods can fast-forward special constant-coefficient tensor-grid problems because both the eigenbasis and the dispersion relation are known explicitly.
- Forcing and nonhomogeneous boundary conditions enter through Duhamel's formula. The propagator may be efficient, but coherent loading of the time-dependent source is an additional input assumption.
- Upwind schemes are stable because they add numerical dissipation. That same dissipative term makes the semidiscrete dynamics nonunitary and leads naturally to dilation methods.

Exercises

Exercise 3.1 (Continuous energy conservation)**.** *For the homogeneous wave equation with homogeneous Dirichlet boundary conditions, differentiate the energy in* (3.3) *and use integration by parts to prove*

$$\frac{d}{dt}E(t) = 0.$$

Exercise 3.2 (Finite-difference wave frequencies)**.** *For the one-dimensional matrix* (3.17)*, compute the eigenvalues and verify that* $\|A_h^{1/2}\| = \Theta(h^{-1})$*.*

Exercise 3.3 (Gradient factorization)**.** *Show that the first-difference matrix* (3.44) *satisfies*

$$D_h^\dagger D_h = h^{-2}\,\mathrm{tridiag}(-1, 2, -1).$$

Exercise 3.4 (Finite-element energy)**.** *Starting from* $M_h\ddot{\boldsymbol{q}}_h + K_h\boldsymbol{q}_h = \boldsymbol{0}$*, prove the finite element energy identity* (3.28)*.*

Exercise 3.5 (Fourier-mode rotations)**.** *Derive the periodic dispersion relation* (3.85) *and the mode rotation* (3.88)*.*

Exercise 3.6 (Hermiticity of the wave Hamiltonian)**.** *Verify that* H_h *in* (3.42) *is Hermitian and that its Schrödinger equation is equivalent to* (3.40)*.*

Exercise 3.7 (Upwind dissipation)**.** *For scalar advection with positive velocity, show that upwinding equals a centered difference plus numerical viscosity.*

Exercise 3.8 (Flux splitting)**.** *For symmetric* $A = S\Lambda S^{-1}$*, derive the flux splitting* (3.115) *and the upwind method* (3.116)*.*

Exercise 3.9 (Moment identities)**.** *Verify the moment identities for the Schrödingerization triple in* (3.130)*.*

Exercise 3.10 (Duhamel–LCU)**.** *Use Duhamel's formula to write a quadrature-based LCU approximation for a forced wave equation. Identify which part of the construction is the homogeneous propagator and which part is the source-state preparation.*

Chapter 4

Quantum Algorithms for Parabolic PDEs

4.1 Classical heat discretizations

Parabolic equations are the dissipative counterpart of the elliptic and hyperbolic problems considered in Chapters 2 and 3. We use the heat equation as the model problem:

$$u_t(\boldsymbol{x}, t) - \nabla \cdot \big(a(\boldsymbol{x})\nabla u(\boldsymbol{x}, t)\big) = f(\boldsymbol{x}, t), \qquad \boldsymbol{x} \in \Omega, \tag{4.1}$$

with initial data

$$u(\boldsymbol{x}, 0) = u_0(\boldsymbol{x}) \tag{4.2}$$

and, unless stated otherwise, homogeneous Dirichlet boundary conditions at the boundary $\partial\Omega$. The coefficient $a(\boldsymbol{x})$ is the heat conductivity, or diffusion coefficient. In this chapter we take it to be scalar and uniformly positive,

$$0 < a_{\min} \leq a(\boldsymbol{x}) \leq a_{\max} < \infty, \tag{4.3}$$

although many of the ideas extend to symmetric positive-definite matrix-valued coefficients. We write the continuous spatial operator as

$$L = -\nabla \cdot (a\nabla), \tag{4.4}$$

using the same notation as in Chapter 2. When $a(\boldsymbol{x}) = c(\boldsymbol{x})^2$, this is the same self-adjoint positive operator that appears in the second-order wave equation

$$u_{tt} + Lu = 0. \tag{4.5}$$

The difference between the three problem classes is which matrix function of L is applied. Elliptic problems use L^{-1}, wave problems use oscillatory functions of $L^{1/2}$, and heat problems use the contraction semigroup

$$u(t) = e^{-tL}u_0 \tag{4.6}$$

when $f = 0$.

For $f = 0$, multiplying (4.1) by u and integrating by parts gives the energy identity

$$\frac{1}{2}\frac{d}{dt}\|u(t)\|^2_{L^2(\Omega)} + \int_\Omega \nabla u(\boldsymbol{x},t)^T a(\boldsymbol{x})\nabla u(\boldsymbol{x},t)\,d\boldsymbol{x} = 0. \tag{4.7}$$

Thus the L^2 norm is nonincreasing. Moreover, since the dissipation term weights gradients, high-frequency components are damped more rapidly than low-frequency components. This smoothing is the main numerical feature of the heat equation, and it will play both sides of the ledger in this chapter. It is helpful for approximation and spectral filtering: a decaying, nonoscillatory filter admits short polynomial and Gaussian representations, which is ultimately why the coherent costs below scale like $\sqrt{T}/h$ rather than T/h^2. But it is also the source of a quantum output issue: preparing the normalized state proportional to $e^{-tL}u_0$ can be expensive if the norm of the physical solution has decayed significantly. This tension—the same decay that makes the *operator* cheap makes the normalized *state* expensive—organizes the whole chapter, and it is resolved only at the level of the output model, by asking for observables rather than states.

The organization is as follows. This section reviews the classical discretizations, following the finite-difference and finite-element analysis of parabolic problems in standard texts such as [53, 84, 82]. We use Crank–Nicolson as a fully discrete benchmark, and we also set up the semidiscrete systems, because these are the objects most naturally fed to QSVT, Gaussian dilation, and Hamiltonian-simulation based primitives. The remaining sections then examine four quantum routes to the same semigroup: a history-state QLSA built on Crank–Nicolson (Section 4.2), a direct QSVT implementation of the exponential filter (Section 4.3), Gaussian dilation through the first-order factor together with observable-driven output (Section 4.4), and QFT-based spectral implementations with positive-time smoothing (Section 4.6).

4.1.1 Finite-difference discretization

On $\Omega = (0, L_{\text{box}})$, we again consider the grid points defined in (2.8). For the constant-coefficient equation $u_t = u_{xx} + f$, the standard second-order centered finite-difference method in space gives

$$\dot{\boldsymbol{u}}_h(t) = -A_h\boldsymbol{u}_h(t) + \boldsymbol{f}_h(t), \qquad A_h = -\Delta_h. \tag{4.8}$$

Here A_h is the same positive tridiagonal matrix introduced in (2.11); in several dimensions it is the Kronecker-sum matrix in (2.21). Hence

$$A_h = A_h^\dagger \succeq 0, \qquad \|A_h\| = \Theta(h^{-2}). \tag{4.9}$$

The semidiscrete solution is simply expressed as,

$$\boldsymbol{u}_h(t) = e^{-tA_h}\boldsymbol{u}_h(0). \tag{4.10}$$

For variable conductivity, the conservative finite-difference discretization is obtained by approximating the flux au_x at cell edges. Let

$$a_{j+1/2} \approx a(x_{j+1/2}). \tag{4.11}$$

Then

$$(A_h\boldsymbol{u})_j = \frac{1}{h^2}\left[a_{j+1/2}(u_j - u_{j+1}) + a_{j-1/2}(u_j - u_{j-1})\right], \qquad j = 1,\dots,N_h, \tag{4.12}$$

with boundary values inserted according to the boundary condition. This is exactly the finite-difference analogue of the weak form: first compute a finite-difference gradient, multiply by the edge conductivity, and then take a negative discrete divergence. The resulting matrix is symmetric positive semidefinite for nonnegative $a_{j+1/2}$, and positive definite under homogeneous Dirichlet boundary conditions. In this notation,

$$A_h = G_h^\dagger W_h G_h, \tag{4.13}$$

where G_h is the first-difference matrix and W_h is the positive diagonal matrix of edge conductivities, matching the notation used for wave equations. Equivalently, after replacing G_h by $W_h^{1/2} G_h$, this has the simple form $A_h = G_h^\dagger G_h$. This is the same divergence-gradient structure used for elliptic problems in Chapter 2, and it is worth noting now: the factorization will be leveraged in Section 4.4.

Let $\boldsymbol{I}_h u(x,t)$ denote the vector of nodal samples of the exact solution, and use the weighted discrete norm $\|\cdot\|_{\ell_h^2}$ introduced in (3.12). For sufficiently smooth solutions, stability of e^{-tA_h} and the $O(h^2)$ consistency of the centered Laplacian yield the representative semidiscrete estimate

$$\max_{0\le t\le T} \|\boldsymbol{u}_h(t) - \boldsymbol{I}_h u(\cdot,t)\|_{\ell^2} \le Ch^2 \left(\|u_0\|_{H^4(\Omega)} + \int_0^T \|u_t(x,s)\|_{H^4(\Omega)}\, ds \right). \tag{4.14}$$

The exact regularity norm is not important here; the important point is that spatial discretization produces a second-order approximation to the heat semigroup.

To integrate (4.8) in time, the operator scale (4.9) matters: the semidiscrete system can be stiff, and an explicit method such as forward Euler is stable only under the restrictive step condition $k = O(h^2)$. The standard remedy is an implicit, unconditionally stable method, and Crank–Nicolson is the second-order representative. Now let $t_n = nk$ with stepsize k and denote the fully discrete Crank–Nicolson iterates by $\boldsymbol{u}_h^n$. Crank–Nicolson applied to (4.8) is

$$\left(I + \frac{k}{2} A_h\right) \boldsymbol{u}_h^{n+1} = \left(I - \frac{k}{2} A_h\right) \boldsymbol{u}_h^n + k \boldsymbol{f}_h^{n+1/2}. \tag{4.15}$$

The local truncation error of Crank–Nicolson for a smooth solution of the semidiscrete ODE is $O(k^3)$ per step, and the centered spatial discretization has local error $O(h^2)$. Each step of Crank–Nicolson involves a rational function of A_h. The eigenvalues, which determine the homogeneous amplification factor and given by

$$r(\lambda) = \frac{1 - k\lambda/2}{1 + k\lambda/2}, \qquad |r(\lambda)| \le 1, \qquad \text{since} \quad \lambda \ge 0, \tag{4.16}$$

is bounded by one, the method is unconditionally stable in the discrete L^2 norm. This also means that the discrete approximation inherits the contraction in time.

Stability converts the accumulated local truncation errors into the global second-order estimate

$$\|\boldsymbol{u}_h^n - \boldsymbol{I}_h u(t_n)\|_{\ell_h^2} \le C_T (h^2 + k^2), \qquad 0 \le t_n \le T. \tag{4.17}$$

Thus one may choose $k = O(h)$ to balance the spatial and temporal errors, rather than the $k = O(h^2)$ restriction required by an explicit scheme. The price of this freedom is that each step of (4.15) requires solving a linear system with the matrix $I + \frac{k}{2} A_h$. Classically, this is the familiar trade of stiff time integration. It is also worth remembering for what follows: when Section 4.2 revisits Crank–Nicolson as one global linear system, these per-step solves are exactly what gets stacked into the quantum linear-system formulation.

4.1.2 Finite-element semidiscretization

Let $V_h \subset H_0^1(\Omega)$ be the same conforming finite-element space used in Section 2.1.3. The weak heat equation is

$$(u_t(\cdot,t),v) + a(u(\cdot,t),v) = (f(t),v), \qquad \forall v \in H_0^1(\Omega), \tag{4.18}$$

where the bilinear form is the elliptic form from (2.25).

Writing

$$u_h(\boldsymbol{x},t) = \sum_{j=1}^{N_h} (\boldsymbol{q}_h(t))_j \phi_j(\boldsymbol{x}), \tag{4.19}$$

and enforcing the weak form against each basis function $v = \phi_i$ yields the semidiscrete finite-element equations

$$M_h \dot{\boldsymbol{q}}_h(t) + K_h \boldsymbol{q}_h(t) = \boldsymbol{b}_h(t). \tag{4.20}$$

The mass matrix M_h, stiffness matrix K_h, and load vector $\boldsymbol{b}_h(t)$ are the same finite-element objects introduced in (2.28) and (2.30). For the homogeneous problem,

$$\frac{1}{2}\frac{d}{dt}(\boldsymbol{q}_h(t)^\dagger M_h \boldsymbol{q}_h(t)) + \boldsymbol{q}_h(t)^\dagger K_h \boldsymbol{q}_h(t) = 0. \tag{4.21}$$

This is the discrete counterpart of (4.7).

As in the elliptic and wave chapters, we can also introduce mass-normalized coordinates, so that the generator becomes a single Hermitian matrix acting on amplitudes:

$$\boldsymbol{y}_h(t) = M_h^{1/2}\boldsymbol{q}_h(t), \qquad A_h = M_h^{-1/2} K_h M_h^{-1/2}, \qquad \boldsymbol{f}_h(t) = M_h^{-1/2}\boldsymbol{b}_h(t). \tag{4.22}$$

Then

$$\dot{\boldsymbol{y}}_h(t) = -A_h \boldsymbol{y}_h(t) + \boldsymbol{f}_h(t), \qquad A_h = A_h^\dagger \succeq 0. \tag{4.23}$$

This is the parabolic analogue of the normalized elliptic system (2.32) and the normalized wave system in Chapter 3. For quasi-uniform meshes and fixed polynomial degree, we still have

$$\|A_h\| = \Theta(h^{-2}). \tag{4.24}$$

Thus finite differences and finite elements arrive at the same abstract object: a semidiscrete contraction generated by a Hermitian positive matrix of norm $\Theta(h^{-2})$. Everything quantum in this chapter starts from this object.

For continuous piecewise-linear elements and sufficiently smooth solutions, standard Galerkin theory for parabolic equations gives, for compatible initial data,

$$\|u_h(\cdot,t) - u(\cdot,t)\|_{L^2(\Omega)} \le Ch^2 \left(\|u_0\|_{H^2(\Omega)} + \int_0^t \|u_t(\cdot,s)\|_{H^2(\Omega)}\, ds \right). \tag{4.25}$$

More refined estimates use the analytic smoothing of the parabolic semigroup; we return to this point in Subsection 4.6.1.

Remark 4.1 (Mass normalization and mass lumping)**.** *The consistent mass matrix does not prevent a quantum treatment: a block encoding of M_h can be combined with QSVT approximations of $M_h^{-1/2}$ to build a block encoding of $A_h = M_h^{-1/2} K_h M_h^{-1/2}$. This is the most literal version of*

the finite-element discretization. For an introductory presentation, however, mass lumping is often simpler. Replacing M_h by a positive diagonal matrix $M_{L,h}$ gives

$$A_{L,h} = M_{L,h}^{-1/2} K_h M_{L,h}^{-1/2}, \tag{4.26}$$

which preserves the local sparsity pattern of K_h and avoids an additional matrix-function step. This is the same simplification used for elliptic eigenproblems and for wave equations in earlier chapters.

4.1.3 Forcing and boundary data through Duhamel's principle

Both finite differences and finite elements reduce the inhomogeneous problem to

$$\dot{\boldsymbol{y}}_h(t) = -A_h \boldsymbol{y}_h(t) + \boldsymbol{f}_h(t). \tag{4.27}$$

Variation of constants gives

$$\boldsymbol{y}_h(T) = e^{-TA_h} \boldsymbol{y}_h(0) + \int_0^T e^{-(T-s)A_h} \boldsymbol{f}_h(s)\, ds. \tag{4.28}$$

After quadrature, the source contribution becomes

$$\int_0^T e^{-(T-s)A_h} \boldsymbol{f}_h(s)\, ds \approx \sum_{m=1}^{N_q} w_m e^{-(T-s_m)A_h} \boldsymbol{f}_h(s_m). \tag{4.29}$$

This expression has the form of an LCU of homogeneous heat propagators applied to time-dependent source vectors. The word "LCU" hides an access assumption: one must coherently prepare the quadrature index, the weights, and normalized source states $|\boldsymbol{f}_h(s_m)\rangle = \boldsymbol{f}_h(s_m)/\|\boldsymbol{f}_h(s_m)\|$, with the norms loaded into the LCU amplitudes, or otherwise embed the source into an autonomous enlarged system. The same issue appeared for body forces in the wave chapter. Because the source term enters every route in this uniform way, the rest of the chapter concentrates on the homogeneous propagator e^{-TA_h}; the reader should keep in mind that each construction below can be wrapped in a Duhamel LCU of exactly this form.

4.2 Crank–Nicolson as a quantum linear system

The previous section treated Crank–Nicolson as a classical time integrator. There is also a quantum linear-system viewpoint: stack all time levels into a single vector and solve one large sparse linear system. The appeal is conceptual economy—time evolution becomes a single linear-algebra object, and the QLSA machinery of Chapter 2 applies verbatim. This history-state or clock-state idea goes back to quantum algorithms for linear ODEs by Berry and collaborators and appears in refined form in later work, including Krovi's ODE algorithm [11, 13, 52]. It is also the most general route in this chapter: nothing in it uses self-adjointness or positivity, which is why it remains available for the nonnormal problems mentioned in the outlook. The question we pursue here is what this generality costs for the heat equation.

For the homogeneous recurrence, define

$$D_+ = I + \frac{k}{2} A_h, \qquad D_- = I - \frac{k}{2} A_h. \tag{4.30}$$

Let $\boldsymbol{y}_h^n$ denote the mass-normalized Crank–Nicolson iterate. Stacking

$$\boldsymbol{Y} = \begin{bmatrix} \boldsymbol{y}_h^0 \\ \boldsymbol{y}_h^1 \\ \vdots \\ \boldsymbol{y}_h^{N_t} \end{bmatrix}, \qquad N_t k = T, \tag{4.31}$$

gives the block lower-bidiagonal system

$$\mathcal{L}_{\mathrm{CN}} \boldsymbol{Y} = \begin{bmatrix} \boldsymbol{y}_h^0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \qquad \mathcal{L}_{\mathrm{CN}} = \begin{bmatrix} I & & & \\ -D_- & D_+ & & \\ & -D_- & D_+ & \\ & & \ddots & \ddots \end{bmatrix}. \tag{4.32}$$

If A_h is sparse, then $\mathcal{L}_{\mathrm{CN}}$ is sparse on the product of the clock register and the spatial register. A QLSA can be implemented by block-encoding this matrix and applying the inverse filter described in (1.111) of Chapter 1.

The conditioning of (4.32) reflects both time and space. A quick way to see the two contributions is to look at the block forward substitution. The matrix itself has block norm

$$\|\mathcal{L}_{\mathrm{CN}}\| \le O(\|D_+\| + \|D_-\| + 1) = O(1 + k\|A_h\|).$$

On the other hand, solving the homogeneous recurrence gives

$$\boldsymbol{y}_h^n = R_k^n \boldsymbol{y}_h^0, \qquad R_k := D_+^{-1} D_-.$$

Since $A_h \succeq 0$, the eigenvalues of R_k are

$$r_k(\lambda) = \frac{1 - k\lambda/2}{1 + k\lambda/2}, \qquad |r_k(\lambda)| \le 1,$$

so the propagator part is contractive. The inverse of the block lower-triangular clock matrix nevertheless contains a causal sum over $N_t + 1$ time levels. This gives the simple estimate

$$\kappa(\mathcal{L}_{\mathrm{CN}}) = O\big((N_t + 1)(1 + k\|A_h\|)\big). \tag{4.33}$$

This is the same type of evolution-norm bookkeeping that appears in quantum algorithms for linear ODEs, including the analyses of Berry et al. and Krovi: the clock system sees both the accumulated time interval and the scale of the linear generator.

One may formally left-precondition each time-step equation by D_+^{-1}. Then the recurrence becomes

$$\boldsymbol{y}_h^{n+1} = R_k \boldsymbol{y}_h^n, \qquad R_k = D_+^{-1} D_-,$$

and the clock matrix has $O(N_t)$ conditioning because R_k is a contraction. But this statement has moved the spatial difficulty into the preconditioner: implementing D_+^{-1} is itself a linear-system problem with

$$\kappa(D_+) = \frac{1 + k\lambda_{\max}(A_h)/2}{1 + k\lambda_{\min}(A_h)/2} = O(1 + k\|A_h\|).$$

Thus a preconditioned clock description is analytically cleaner, but the parabolic scale $\|A_h\| = \Theta(h^{-2})$ has not disappeared. The lesson is a kind of conservation of difficulty: the clock formulation lets one shift the parabolic stiffness between the clock matrix and its preconditioner, but no purely algebraic rearrangement removes it.

The QLSA output is a normalized history state proportional to

$$\sum_{n=0}^{N_t} |n\rangle_{\text{clock}} \, |\boldsymbol{y}_h^n\rangle \, . \tag{4.34}$$

For the homogeneous heat equation, Crank–Nicolson is unconditionally stable and the amplification factor satisfies $|r_k(\lambda)| \le 1$. Therefore

$$\|\boldsymbol{y}_h^n\|_2 \le \|\boldsymbol{y}_h^0\|_2, \qquad n = 0, \dots, N_t.$$

The largest time slice is the initial one, not an unknown intermediate time. Define the discrete output ratio

$$g_T^{\text{CN}} := \frac{\|\boldsymbol{y}_h^0\|_2}{\|\boldsymbol{y}_h^{N_t}\|_2}. \tag{4.35}$$

Then the probability of measuring the final clock value satisfies

$$p_{N_t} = \frac{\|\boldsymbol{y}_h^{N_t}\|_2^2}{\sum_{n=0}^{N_t} \|\boldsymbol{y}_h^n\|_2^2} \ge \frac{1}{(N_t+1)(g_T^{\text{CN}})^2}. \tag{4.36}$$

The bound exposes two obstructions, and they are of different kinds. The factor $N_t + 1$ is clock dilution: the answer is spread uniformly over the time register. The factor $(g_T^{\text{CN}})^2$ is physical decay: the final-time slice is genuinely small. Padding the clock with additional copies of the final state can make the clock probability constant [52], but the amount of padding grows with the same norm ratio; padding cures the dilution, not the decay. In the limit $k \to 0$, g_T^{CN} is the discrete analogue of the heat output ratio g_T in (4.50). Even though the heat semigroup is stable,

$$\sup_{0 \le t \le T} \|e^{-tA_h}\| \le 1, \qquad \|A_h\| = \Theta(h^{-2}), \tag{4.37}$$

the final norm may be much smaller than the initial norm.

Combining the clock conditioning, spatial scale, and output ratio gives the schematic history-state cost

$$\widetilde{O}\left(g_T^{\text{CN}} \, T \|A_h\| \, \text{polylog}(1/\epsilon)\right) = \widetilde{O}\left(g_T^{\text{CN}} \, \frac{T}{h^2} \, \text{polylog}(1/\epsilon)\right), \tag{4.38}$$

up to sparsity, state-preparation, and clock-padding factors. This approach is conceptually important and applies to nonnormal or inhomogeneous ODEs. For a self-adjoint heat semigroup, however, QSVT and Gaussian dilation exploit positivity and smoothing more directly, and the next two sections show that they replace the factor T/h^2 by its square root.

4.3 QSVT implementation of the heat semigroup

The homogeneous semidiscrete heat problem asks us to apply

$$E_T := e^{-TA_h}, \qquad A_h = A_h^\dagger \succeq 0. \tag{4.39}$$

Suppose that A_h/α_A has a normalized Hermitian block encoding, with $\alpha_A \geq \|A_h\|$. Set

$$X_h := A_h/\alpha_A, \qquad \tau := T\alpha_A. \tag{4.40}$$

Then $\mathrm{spec}(X_h) \subset [0,1]$, and

$$e^{-TA_h} = e^{-TX_h}.$$

At first sight this looks like the scalar function

$$x \mapsto e^{-\tau x}, \qquad x \in [0,1].$$

There is a subtle but important QSVT point here. The physical spectrum of X_h lies in $[0,1]$, but an admissible QSVT polynomial must be bounded on the full signal interval $[-1,1]$. The function e^{-Tx} is bounded by one on $[0,1]$, but it grows to e^{τ} on the negative half of the signal interval. Thus one cannot use an arbitrary Chebyshev approximation on $[0,1]$ without controlling its extension to $[-1,1]$.

The standard remedy [33] is to reverse the normalized spectral variable. Define

$$B_h := I - X_h = I - \frac{A_h}{\alpha_A}. \tag{4.41}$$

No independent oracle for B_h is assumed: starting from the Hermitian block encoding of X_h, the affine map $x \mapsto 1-x$ is incorporated by the standard affine interval transformation for block encodings, with no change in the asymptotic query complexity. Now we have $\mathrm{spec}(B_h) \subset [0,1]$,

$$e^{-TA_h} = e^{-T(I-B_h)}. \tag{4.42}$$

The relevant scalar function is now

$$f(x) = e^{-T(1-x)}, \qquad x \in [-1,1]. \tag{4.43}$$

This shifted exponential satisfies $e^{-2T} \leq f(x) \leq 1 \quad$ for all $x \in [-1,1]$. It is therefore compatible with the global boundedness requirement of QSVT.

The standard QSVT machinery [33, Corollary 64] produces a real polynomial $p_{\tau,\epsilon}$ satisfying

$$|p_{T,\epsilon}(x)| \leq 1 \quad (x \in [-1,1]), \qquad |p_{T,\epsilon}(x) - e^{-T(1-x)}| \leq \epsilon \quad (x \in [0,1]). \tag{4.44}$$

This is the same square-root exponential approximation used in the QSVT framework for imaginary-time evolution and Gibbs filtering [33]. Approximation-theoretic refinements give closely related optimal degree estimates [1]. The degree obeys

$$m = O\left(\sqrt{\max\{T, \log(1/\epsilon)\}\log(1/\epsilon)}\right), \tag{4.45}$$

or, in the diffusion-dominated regime $\tau \gtrsim \log(1/\epsilon)$,

$$m = \widetilde{O}\left(\sqrt{T\log(1/\epsilon)}\right). \tag{4.46}$$

Where does the square root come from? A useful heuristic is the endpoint clustering of Chebyshev oscillations: a polynomial of degree m, bounded by one on $[-1,1]$, can resolve features

of width $\Theta(1/m^2)$ near the ends of the interval, because the extrema of T_m cluster quadratically there. The heat filter $e^{-\tau x}$ is a boundary layer of width $\Theta(1/\tau)$ at the endpoint $x = 0$: it drops from 1 to ϵ over a spectral window of length $\log(1/\epsilon)/\tau$ and is essentially flat afterwards. Equating the two scales, $1/m^2 \sim 1/\tau$, gives $m \sim \sqrt{\tau}$. This heuristic also explains why no such saving occurs for the oscillatory filter e^{-itx} of Hamiltonian simulation: that function has features of size $1/t$ *everywhere* in the interval, not only in an endpoint layer, and its polynomial degree is therefore proportional to t.

Therefore QSVT produces a block encoding of $E_T = e^{-TA_h}$ using

$$Q_{\mathrm{QSVT}} = \widetilde{O}\left(\sqrt{T\alpha_A \log(1/\epsilon)} + \log(1/\epsilon)\right) \tag{4.47}$$

queries to the block encoding of A_h/α_A. Since $\alpha_A = \Theta(h^{-2})$ for second-order finite differences and mass-normalized finite elements,

$$Q_{\mathrm{QSVT}} = \widetilde{O}\left(\frac{\sqrt{T}}{h}\sqrt{\log(1/\epsilon)} + \log(1/\epsilon)\right). \tag{4.48}$$

This square-root scale comes from the fact that the heat semigroup is a decaying, nonoscillatory filter. This also explains why the parabolic problem behaves differently from generic Hamiltonian simulation, where the oscillatory function e^{-itx} normally has degree proportional to t.

The estimate above is a coherent operator cost. It is not automatically the cost of preparing the normalized final heat state. If

$$|\boldsymbol{y}_0\rangle = \frac{\boldsymbol{y}_h(0)}{\|\boldsymbol{y}_h(0)\|_2}, \qquad p_T = \|E_T\,|\boldsymbol{y}_0\rangle\,\|_2^2, \tag{4.49}$$

then postselecting the block-encoding ancilla succeeds with probability p_T. Preparing the normalized state $E_T\,|\boldsymbol{y}_0\rangle\,/\|E_T\,|\boldsymbol{y}_0\rangle\,\|$ therefore requires an additional factor

$$g_T := p_T^{-1/2} = \frac{\|\boldsymbol{y}_h(0)\|_2}{\|\boldsymbol{y}_h(T)\|_2}. \tag{4.50}$$

Under homogeneous Dirichlet boundary conditions, the smallest eigenvalue is bounded below, $\lambda_{1,h} = \Theta(1)$, so long-time heat flow can make g_T exponentially large in T. This is the central distinction between implementing the heat operator and producing a normalized heat-solution state. Section 4.5 shows how to sidestep the factor g_T entirely when the desired output is a scalar observable rather than the state itself.

4.4 First-order Hermitian dilation and Gaussian transmutation

Recall that our discretization in space has the factorization property

$$A_h = G_h^\dagger G_h, \tag{4.51}$$

as in the elliptic factorization (2.41), and $\|G_h\| = \Theta(h^{-1})$ for standard second-order discretizations.

As in the elliptic and wave chapters, the Hermitian dilation is the right device:

$$\mathcal{D}(G_h) = \begin{bmatrix} 0 & G_h^\dagger \\ G_h & 0 \end{bmatrix}. \tag{4.52}$$

Then

$$\mathcal{D}(G_h)^2 = \begin{bmatrix} A_h & 0 \\ 0 & G_h G_h^\dagger \end{bmatrix}, \tag{4.53}$$

so A_h appears as the upper-left block: squaring the first-order dilation reproduces the second-order operator, just as squaring the wave Hamiltonian recovered the stiffness matrix in Chapter 3. The dilation is Hermitian, so $e^{-is\mathcal{D}(G_h)}$ is a unitary that Hamiltonian simulation can implement. What is still missing is a way to assemble the *decaying* function $e^{-T\mathcal{D}(G_h)^2}$ out of these oscillatory unitaries. Classical Fourier analysis gives the simple recipe: the Gaussian is, up to scaling, its own Fourier transform. The key observation is the Gaussian Fourier identity

$$e^{-Tx^2} = \frac{1}{2\sqrt{\pi T}} \int_{-\infty}^{\infty} e^{-s^2/(4T)} e^{-isx}\, ds \tag{4.54}$$

therefore gives, by the spectral theorem,

$$e^{-T\mathcal{D}(G_h)^2} = \frac{1}{2\sqrt{\pi T}} \int_{-\infty}^{\infty} e^{-s^2/(4T)} e^{-is\mathcal{D}(G_h)}\, ds. \tag{4.55}$$

The upper-left block of the left-hand side is e^{-TA_h}. Thus heat flow is represented as a Gaussian average of unitary wave evolutions generated by the first-order operator $\mathcal{D}(G_h)$. Formulas of this type are classical in PDE theory, where they express the heat kernel as a Gaussian average of wave kernels; the quantum literature has rediscovered and operationalized them. This construction is closely related to the Gaussian-LCHS construction of Kharazi et al. and to the transmutation approach of Jin, Ma, and Zuazua [51, 48]. It is also an instance of the broader dilation viewpoint introduced in Chapter 3: after diagonalizing or quadraturing the ancillary Gaussian variable, the heat semigroup is represented as an LCU of unitary evolutions, much like (3.132).

Truncating and discretizing the integral gives an LCU

$$e^{-T\mathcal{D}(G_h)^2} \approx \sum_{m=1}^{N_q} c_m(T) e^{-is_m \mathcal{D}(G_h)}, \tag{4.56}$$

where

$$\sum_m |c_m(T)| = O(1), \qquad \max_m |s_m| = O\left(\sqrt{T\log(1/\epsilon)}\right). \tag{4.57}$$

These two scales tell the whole story. The weights have $O(1)$ total mass, so the LCU incurs no normalization penalty, and the only remaining postselection loss is the physical decay of the heat solution. The longest Hamiltonian evolution in the integral runs only to time $O(\sqrt{T\log(1/\epsilon)})$, because that is where the Gaussian window has decayed below the tolerance. The diffusive square root, which appeared in Section 4.3 as a polynomial degree, reappears here as the width of a Gaussian. If $\mathcal{D}(G_h)/\alpha_G$ has a block encoding with $\alpha_G = \Theta(h^{-1})$, Hamiltonian simulation and LCU yield

$$Q_{\text{Gauss}} = \widetilde{O}\left(\alpha_G\sqrt{T\log(1/\epsilon)} + \log(1/\epsilon)\right) = \widetilde{O}\left(\frac{\sqrt{T}}{h}\sqrt{\log(1/\epsilon)}\right). \tag{4.58}$$

Thus direct QSVT and Gaussian dilation have the same leading coherent scaling. The advantage of Gaussian dilation is not an asymptotic improvement under identical access assumptions; it is an alternative realization that may be natural when the first-order operator G_h or its dilation is easier to simulate than the second-order operator A_h is to block encode.

A heuristic comparison with classical Crank–Nicolson. It is useful to compare (4.58) with a classical Crank–Nicolson computation while keeping the physical parameters visible. On a d-dimensional domain of characteristic side length L_{box}, a mesh with spacing h has

$$\left(\frac{L_{\mathrm{box}}}{h}\right)^d$$

spatial degrees of freedom, up to constants depending on the element type, boundary treatment, and geometry.

A Crank–Nicolson step for the mass-normalized heat equation has the form

$$\left(I + \frac{k}{2}A_h\right)\boldsymbol{y}_h^{n+1} = \left(I - \frac{k}{2}A_h\right)\boldsymbol{y}_h^{n}.$$

The method is unconditionally stable, but accuracy still constrains the time step. In nondimensional variables, the standard second-order estimate has the form

$$O(h^2 + k^2).$$

Thus a balanced choice is $k = \Theta(h)$, giving $O(T/h)$ time steps. If one keeps physical units, the same statement should be read after nondimensionalizing time by the diffusive scale $L_{\mathrm{box}}^2/a_{\mathrm{ref}}$.

The cost of each Crank–Nicolson step is the cost of solving

$$\left(I + \frac{k}{2}A_h\right)\boldsymbol{z} = \boldsymbol{r}.$$

Since

$$\|A_h\| = \Theta(a_{\max}h^{-2}),$$

the step matrix has condition number bounded schematically by

$$\kappa\left(I + \frac{k}{2}A_h\right) \lesssim 1 + k\|A_h\| = O\left(1 + \frac{k\,a_{\max}}{h^2}\right).$$

Thus a linear solver still feels the parabolic spatial scale. With multigrid, BPX, or another near-optimal elliptic preconditioner, each implicit solve can be performed in essentially

$$O\left(\left(\frac{L_{\mathrm{box}}}{h}\right)^d\right)$$

work, up to logarithmic factors and coefficient-dependent constants. With $k = \Theta(h)$ in nondimensional units, the full classical Crank–Nicolson computation therefore costs

$$\widetilde{O}\left(\frac{T}{h}\left(\frac{L_{\mathrm{box}}}{h}\right)^d\right) = \widetilde{O}\left(T\,L_{\mathrm{box}}^d\,h^{-d-1}\right)$$

in nondimensional variables.

The quantum cost in (4.58) is of a different type. It is the coherent cost of applying the heat semigroup to an amplitude-encoded state:

$$Q_{\text{heat}} = \widetilde{O}\left(\sqrt{T\|A_h\|}\right) = \widetilde{O}\left(\frac{\sqrt{T}}{h}\right).$$

4.5 Direct estimation of a PDE quantity of interest

As alluded to in previous sections, for dissipative dynamics, producing the normalized final state is often not the actual computational goal. A PDE calculation usually asks for a scalar output: a heat content, a regional average, a reaction rate, or a weighted observable. Let

$$\mathcal{Q}(T) = \boldsymbol{\ell}_h^\dagger \boldsymbol{y}_h(T) = \boldsymbol{\ell}_h^\dagger e^{-TA_h} \boldsymbol{y}_h(0), \tag{4.59}$$

be the linear quantity expressed at the discrete level.

Introduce normalized states

$$|\boldsymbol{\psi}\rangle = \frac{\boldsymbol{y}_h(0)}{\|\boldsymbol{y}_h(0)\|_2}, \qquad |\boldsymbol{\phi}\rangle = \frac{\boldsymbol{\ell}_h}{\|\boldsymbol{\ell}_h\|_2}. \tag{4.60}$$

Then

$$\mathcal{Q}(T) = \|\boldsymbol{\ell}_h\|_2 \, \|\boldsymbol{y}_h(0)\|_2 \, \langle\boldsymbol{\phi}| \, e^{-TA_h} \, |\boldsymbol{\psi}\rangle \, . \tag{4.61}$$

The factorization performs a clean division of labor: every mesh- and problem-dependent magnitude is an explicit classical prefactor, known in advance, while the quantum computer is asked only for a dimensionless matrix element of modulus at most one. This is the same lesson emphasized by direct-overlap and observable-driven heat algorithms, and by the end-to-end analysis of Linden, Montanaro, and Shao [59, 51, 63].

Once either QSVT or Gaussian dilation has produced a block encoding of e^{-TA_h}, the matrix element in (4.61) can be estimated by the Hadamard test or by amplitude estimation, as reviewed in Chapter 1. If the target physical additive error is ϵ, then the normalized matrix element $\langle\boldsymbol{\phi}| \, e^{-TA_h} \, |\boldsymbol{\psi}\rangle$ must be estimated to accuracy

$$\frac{\epsilon}{\|\boldsymbol{\ell}_h\|_2 \, \|\boldsymbol{y}_h(0)\|_2}. \tag{4.62}$$

Plain sampling squares this inverse accuracy, while amplitude estimation uses it linearly. If C_E denotes the coherent cost of one block encoding of the heat semigroup, the amplitude-estimation cost is schematically

$$\widetilde{O}\left(C_E \, \frac{\|\boldsymbol{\ell}_h\|_2 \, \|\boldsymbol{y}_h(0)\|_2}{\epsilon}\right), \tag{4.63}$$

plus state-preparation costs for $|\boldsymbol{\psi}\rangle$ and $|\boldsymbol{\phi}\rangle$. Direct sampling replaces the final factor by its square. Crucially, neither approach needs to prepare

$$\frac{e^{-TA_h} \, |\boldsymbol{\psi}\rangle}{\|e^{-TA_h} \, |\boldsymbol{\psi}\rangle \, \|}. \tag{4.64}$$

Therefore direct estimation can avoid the exponentially large g_T factor in (4.50) when the desired output is an additive physical observable. This resolves, at the level of the output model, the tension announced at the start of the chapter: the decay of the solution is absorbed into a small matrix element to be estimated, rather than into a small postselection probability to be amplified.

The work of Linden, Montanaro, and Shao illustrates why this distinction is not cosmetic. For their heat-content benchmark, a QLSA route based on a history-state formulation is not asymptotically faster than the best classical methods, while the strongest quantum improvement comes from applying amplitude estimation to a carefully chosen stochastic representation [63]. In other words, end-to-end heat-equation complexity is controlled not only by the coherent propagation primitive, but also by discretization, normalization, and measurement.

4.6 QFT methods and positive-time spectral smoothing

For constant coefficients on periodic tensor-product grids, the Fourier basis provides an explicit diagonalization. Chapter 2 already introduced this structure for Poisson-type problems. With $\mathcal{F}_h = F_{N_{\mathrm{cell}}}^{\otimes d}$,

$$A_h = \mathcal{F}_h^\dagger \Lambda_h \mathcal{F}_h, \qquad \Lambda_h = \mathrm{diag}(\lambda_{\boldsymbol{k}}), \tag{4.65}$$

and, by (2.75),

$$\lambda_{\boldsymbol{k}} = \frac{4}{h^2} \sum_{r=1}^{d} \sin^2\left(\frac{\pi k_r}{N_{\mathrm{cell}}}\right), \qquad \boldsymbol{k} = (k_1, \ldots, k_d). \tag{4.66}$$

Here N_{cell} denotes the number of periodic grid points per coordinate direction.

Thus each Fourier coefficient decays independently:

$$\widehat{y}_{\boldsymbol{k}}(T) = e^{-T\lambda_{\boldsymbol{k}}} \widehat{y}_{\boldsymbol{k}}(0). \tag{4.67}$$

Equivalently,

$$e^{-TA_h} = \mathcal{F}_h^\dagger \,\mathrm{diag}\left(e^{-T\lambda_{\boldsymbol{k}}}\right) \mathcal{F}_h. \tag{4.68}$$

A QFT implementation applies $\mathcal{F}_h$, computes $\lambda_{\boldsymbol{k}}$ coherently by reversible arithmetic, performs a controlled rotation with amplitude $e^{-T\lambda_{\boldsymbol{k}}}$, uncomputes the arithmetic, and applies the inverse QFT. Explicit Fourier-space circuits for heat, advection, wave, and Poisson equations are developed in [68].

The formula (4.67) makes the parabolic smoothing visible. Low frequencies, for which $\lambda_{\boldsymbol{k}} T \ll 1$, are transmitted almost unchanged. High frequencies, for which $\lambda_{\boldsymbol{k}} T \gg 1$, are suppressed. This is the same physical mechanism behind the square-root complexity in QSVT and Gaussian dilation. The largest discrete frequency is $\lambda_{\max}(A_h) = \Theta(h^{-2})$, so the dimensionless diffusion time is $\tau = T\|A_h\| = \Theta(T/h^2)$. Bounded polynomial or Gaussian representations of the heat filter have degree or propagation time

$$\widetilde{O}(\sqrt{\tau}) = \widetilde{O}(\sqrt{T}/h),$$

rather than $\widetilde{O}(\tau) = \widetilde{O}(T/h^2)$. In this sense, heat smoothing gives a quadratic improvement over a naive time-marching or Taylor-series scale.

It is useful to separate two favorable effects. First, the QFT diagonalizes the operator, so the matrix action becomes scalar arithmetic on the mode register. If the sine and exponential in (4.66) are computed explicitly and reversibly, the coherent gate count for the diagonal filter can be polylogarithmic in the number of grid points, apart from the arithmetic precision and postselection. Second, even when the diagonal filter is implemented through a generic polynomial, Chebyshev, or Gaussian construction, the smoothing of the heat semigroup gives the $\widetilde{O}(\sqrt{T}/h)$ scale. The first effect uses separability and explicit spectral formulas; the second effect is a general property of the decaying exponential.

The price of normalized-state preparation remains. If

$$|\boldsymbol{y}_0\rangle = \sum_{\boldsymbol{k}} \widehat{y}_{\boldsymbol{k}} \, |\boldsymbol{k}\rangle \tag{4.69}$$

in Fourier space, then the postselection probability of the diagonal heat filter is

$$p_T^{\mathrm{QFT}} = \sum_{\boldsymbol{k}} |\widehat{y}_{\boldsymbol{k}}|^2 e^{-2T\lambda_{\boldsymbol{k}}} = \| e^{-TA_h} \, |\boldsymbol{y}_0\rangle \, \|_2^2. \tag{4.70}$$

Thus QFT diagonalization simplifies the operator implementation, but it does not remove the physical decay of the solution.

4.6.1 Positive-time smoothing and spectral truncation

So far, parabolic smoothing has mostly appeared on the cost side of the ledger, through the decayed output norm. Here it becomes an asset: smoothing reduces the spatial resolution that the problem actually requires, an effect that is visible to PDE analysis but invisible to a purely algebraic condition-number discussion. At positive time, high modes have already been damped. From (4.67), modes satisfying

$$\lambda_{\boldsymbol{k}} \geq \lambda_{\mathrm{cut}} := \frac{1}{T} \log \frac{1}{\epsilon} \tag{4.71}$$

contribute at most ϵ in operator norm. On a periodic grid, the small-frequency expansion of (4.66) gives

$$\lambda_{\boldsymbol{k}} \approx 4\pi^2 |\boldsymbol{k}|^2 / L_{\mathrm{box}}^2.$$

Thus the relevant Fourier window at time T is approximately

$$|\boldsymbol{k}| \lesssim \frac{L_{\mathrm{box}}}{2\pi} \sqrt{\frac{1}{T} \log \frac{1}{\epsilon}}. \tag{4.72}$$

This is the spectral form of parabolic smoothing: the longer the heat equation runs, the fewer high-frequency modes survive above a fixed tolerance.

The same phenomenon appears in finite-element error estimates. In addition to uniform-in-time $O(h^2)$ estimates for smooth solutions, analytic-semigroup smoothing gives a positive-time operator estimate of the form

$$\| E(T) - E_h(T) P_h \|_{L^2 \to L^2} \leq C \min \left\{ 1, \frac{h^2}{T} \right\}, \qquad T > 0, \tag{4.73}$$

where $E(T) = e^{-TL}$, $E_h(T)$ is the semidiscrete heat semigroup, and P_h is typically the L^2 projection onto the finite-element space. Such positive-time estimates are standard in parabolic

finite-element theory [82, Ch. 3]. This positive-time estimate is useful when T is fixed away from zero; it does not justify using a mesh coarser than the physical feature scale or the domain geometry permits.

If the physical output is measured at a fixed positive time, (4.73) may allow a coarser mesh than a uniform-in-time worst-case estimate would suggest. Balancing $h^2/T \approx \epsilon$ gives

$$h \approx \sqrt{T\epsilon}. \tag{4.74}$$

Substituting this into the coherent scale $\sqrt{T}/h$ gives an $O(\epsilon^{-1/2})$ dependence for the semigroup block-encoding part, whereas choosing $h = O(\sqrt{\epsilon})$ from a time-uniform $O(h^2)$ estimate would give the usual $O(\sqrt{T}/\sqrt{\epsilon})$ scaling. Note that the T-dependence has cancelled entirely in the first case: running the equation longer permits a coarser mesh at exactly the rate that offsets the longer diffusion time. The exact conclusion for an end-to-end algorithm still depends on the observable norm, state preparation, and measurement, but the positive-time smoothing estimate is a useful PDE-level improvement that is invisible in a purely algebraic condition-number discussion.

For rectangular domains with separable boundary conditions, the same truncation idea can be implemented with sine or cosine transforms rather than the periodic QFT. This is the parabolic analogue of classical FFT, DST, and DCT solvers for heat equations on boxes.

4.7 Summary and outlook

The heat equation reuses the elliptic spatial operators of Chapter 2, but changes the matrix function from an inverse to a decaying exponential. The main conclusions are:

1. Finite differences and finite elements lead to semidiscrete systems

$$\dot{\boldsymbol{y}}_h = -A_h \boldsymbol{y}_h + \boldsymbol{f}_h(t), \qquad A_h \succeq 0, \qquad \|A_h\| = \Theta(h^{-2}). \tag{4.75}$$

   Crank–Nicolson is the standard second-order fully discrete method; its unconditional stability permits $k = O(h)$ when the spatial error is $O(h^2)$.

2. A Crank–Nicolson history-state QLSA is possible, but its cost retains both the clock length and the parabolic spatial scale. In the self-adjoint heat setting it is generally less favorable than QSVT or Gaussian dilation, which exploit the bounded exponential filter more directly. An open question is whether sharper history-state analyses or revised preconditioned clock systems can reduce this gap for parabolic PDEs with sources and boundary data.

3. Direct QSVT implements the heat semigroup with coherent query complexity

$$\widetilde{O}(\sqrt{T\|A_h\|}) = \widetilde{O}(\sqrt{T}/h), \tag{4.76}$$

   up to logarithmic precision factors. The square-root dependence follows from polynomial approximation of a decaying exponential.

4. Gaussian dilation achieves the same coherent scale by representing heat flow as a Gaussian average of unitary wave evolutions generated by the first-order dilation $\mathcal{D}(G_h)$. It is most attractive when the first-order factor is the natural access model.

5. QFT methods give an especially transparent spectral implementation for constant-coefficient tensor-product problems. They can exploit explicit symbols, sine/cosine transforms, and positive-time spectral truncation, but normalized-state preparation still pays for heat decay.

6. Direct estimation of physical quantities of interest can avoid producing the decayed normalized heat state. This is often the right end-to-end goal for parabolic PDEs.

Several issues remain open.

Source terms and boundary conditions. Source terms and nonhomogeneous boundary data require coherent Duhamel constructions, source-history loading, or enlarged autonomous systems. The main difficulty is not the variation-of-constants formula itself, but the quantum access model for the time-dependent forcing states and their normalization factors.

Nonnormal parabolic problems. Convection–diffusion equations, absorbing boundary conditions, and stabilized discretizations can lead to nonnormal generators. These problems require tools beyond self-adjoint spectral filters, such as dilations, singular-value transformations, pseudospectral estimates, or problem-specific preconditioning.

History-state QLSA versus semigroup filters. The QLSA/history-state approach is broadly applicable because it starts from a standard time discretization such as Crank–Nicolson. However, in its basic form it does not match the coherent complexity of direct QSVT or Gaussian dilation for heat semigroups. It remains open whether sharper conditioning estimates, better preconditioning, or different clock formulations can close this gap.

Positive-time smoothing and observables. The positive-time smoothing estimates used above suggest that, for many quantities of interest, the mesh need not resolve modes that have already been damped below the target tolerance. Extending this observable-driven viewpoint to general finite elements, systems of parabolic equations, and stochastic parabolic models is an important direction for future work.

4.8 Exercises

Exercise 4.1 (Semidiscrete heat operator)**.** *Starting from the matrix A_h in (2.11), derive the semidiscrete heat equation (4.8) and verify $\|A_h\| = \Theta(h^{-2})$.*

Exercise 4.2 (Crank–Nicolson stability)**.** *Derive the Crank–Nicolson amplification factor (4.16) and show that its modulus is at most one. Explain why this unconditional stability leads to a global $O(k^2)$ temporal error for smooth solutions.*

Exercise 4.3 (Finite-element energy)**.** *Starting from $M_h\dot{\boldsymbol{q}}_h + K_h\boldsymbol{q}_h = 0$, derive (4.23) and the energy identity (4.21).*

Exercise 4.4 (Bounded polynomial for the heat semigroup)**.** *Let $A_h \succeq 0$ and suppose that A_h/α_A is block encoded, with spectrum in $[0,1]$. Explain why QSVT requires a polynomial that is bounded on all of $[-1,1]$, not merely on $[0,1]$. Using (4.44), explain why a bounded approximation to e^{-TA_h} has degree $\widetilde{O}(\sqrt{T\alpha_A \log(1/\epsilon)})$.*

Exercise 4.5 (Gaussian identity)**.** *Prove the Gaussian identity (4.54) and use the spectral theorem to derive (4.55).*

Exercise 4.6 (Clock probability)**.** *For the Crank–Nicolson history state (4.34), prove (4.36). How much padding is needed to make the probability of observing the final-time block constant?*

Exercise 4.7 (Positive-time mesh selection)**.** *Use the smoothing estimate (4.73) to choose h so that the spatial error at a fixed time $T > 0$ is at most ϵ. Compare this choice with the time-uniform estimate $O(h^2)$.*

Exercise 4.8 (Direct quantity of interest)**.** *Compare the cost of preparing the normalized final heat state with the cost of directly estimating the quantity of interest in (4.61).*

Chapter 5

A First Look at Nonlinear Problems

5.1 Why nonlinear PDEs require a different viewpoint

The algorithms in previous chapters relied heavily on the linear structure of the PDEs: elliptic equations led to linear systems, hyperbolic equations led to Hamiltonian dynamics, and parabolic equations led to contraction semigroups. Nonlinear partial differential equations do not fit so directly into this picture. The superposition principle fails, stability may depend on the solution itself, shocks or caustics may form, and the appropriate notion of solution may be weak, entropy, viscosity, measure-valued, or probabilistic. Thus one should not expect a single quantum primitive to handle nonlinear PDEs in the same way that Hamiltonian simulation handles a finite-dimensional Schrödinger equation.

There are special exceptions. If a nonlinear evolution can be mapped by an explicit transform to a stable linear problem, then the linear quantum methods of the previous chapters may become directly relevant. The classical Cole–Hopf transformation for viscous Burgers and Hamilton–Jacobi equations is the standard example: after an exponential change of variables, a nonlinear viscous equation becomes a heat equation. More recent work of Jin and Liu uses entropy penalization to generalize this idea to viscosity solutions of convex Hamilton–Jacobi equations, reducing the extraction of point values, gradients, and minima to heat-like linear dynamics [44]. These examples are important because they preserve the physically relevant nonlinear solution concept while giving a linear computational representation.

Such transformations, however, are problem dependent. For a general nonlinear PDE, after appropriate spatial discretization one obtains a nonlinear ODE

$$\frac{d}{dt}\boldsymbol{u}(t) = \Phi(\boldsymbol{u}(t), t), \qquad \boldsymbol{u}(t) \in \mathbb{C}^N, \tag{5.1}$$

and there is no universal linear quantum evolution whose amplitudes equal $\boldsymbol{u}(t)$ for all possible nonlinear vector fields Φ. This chapter therefore presents two elementary linearization viewpoints. They are included mainly as templates, not as general-purpose algorithms.

The first is *Carleman linearization*. It maps polynomial nonlinear dynamics to an infinite-dimensional linear flow on tensor powers, or monomials, of the state. After truncation, the problem becomes a large linear ODE and can be treated by QLSA, quantum linear-ODE solvers, or dilation methods presented in previous chapters. This viewpoint is best interpreted as a route

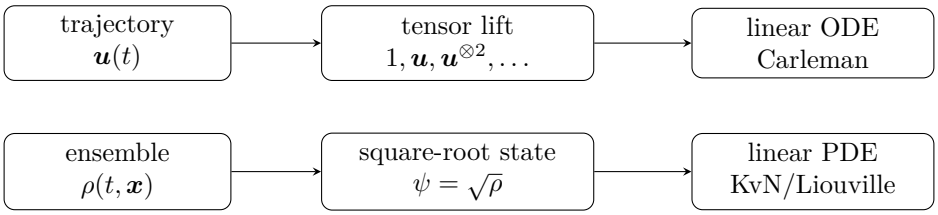

Figure 5.1: Two linear ways to look at nonlinear dynamics. Carleman lifting keeps a single trajectory but enlarges the state by tensor powers. Liouville/KvN lifting moves to probability or observable evolution on phase space.

to *trajectory simulation*: given one initial condition, prepare or estimate the solution at a later time.

The second viewpoint is the *Liouville*, *Koopman*, or *Koopman–von Neumann* formulation. Instead of following one trajectory, one follows a density or an observable on phase space. The nonlinear dynamics becomes a linear, but also infinite dimensional, transport equation for the probability density, or equivalently a Schrödinger-type equation for a wavefunction whose squared modulus is the density. This viewpoint is best interpreted as a route to *probability and ensemble simulation*: compute expectations, correlations, or uncertainty-quantification observables.

Figure 5.1 previews the two routes. They start from different objects—a single trajectory versus an ensemble—and they arrive at different linear problems. Much of this chapter consists of making the middle arrows precise, and of asking, in the spirit of the earlier chapters, what each route costs end to end: state preparation, linear evolution, and the extraction of a physical answer.

For classical nonlinear hyperbolic problems, finite-volume methods, Riemann solvers, limiters, and entropy conditions are central; a standard reference is LeVeque's book [54]. On the quantum side, Carleman linearization has been analyzed for dissipative nonlinear differential equations [64], including reaction–diffusion models [3]. It has been improved for more general linearized dynamics using evolution-norm bounds [52], extended to regimes beyond the traditional dissipative condition [87], and developed further in recent embedding and nonlinear-solver frameworks [40, 25]. The Koopman–von Neumann formulation for quantum simulation of nonlinear classical dynamics was developed in [49]; related Liouville, level-set, and observable-computation methods for nonlinear PDEs are discussed in [46, 42].

The purpose here is therefore not to claim a broad quantum advantage for nonlinear PDEs. Rather, it is to show the reader two linearization mechanisms and to clarify what each mechanism computes.

5.2 From nonlinear PDEs to nonlinear ODEs

A reasonable starting point is semidiscrete approximation, where the continuous PDE is first discretized in space, producing a finite-dimensional nonlinear ODE of the form already displayed in (5.1). For example, a scalar conservation law

$$u_t + f(u)_x = 0 \tag{5.2}$$

may be approximated by a finite-volume method

$$\frac{d}{dt}u_j(t) = -\frac{1}{h}\left(\widehat{f}_{j+1/2}(\boldsymbol{u}(t)) - \widehat{f}_{j-1/2}(\boldsymbol{u}(t))\right), \tag{5.3}$$

where $\widehat{f}$ is a numerical flux. For smooth solutions and simple fluxes, this is a finite-dimensional nonlinear ODE. For discontinuous solutions, however, the numerical flux is part of the mathematical problem: it selects the entropy solution and controls spurious oscillations [54].

As a second, closely related example, the viscous Burgers equation

$$u_t + uu_x = \nu u_{xx} \tag{5.4}$$

may lead to a semidiscrete system of the form

$$\dot{\boldsymbol{u}} = -\nu A_h \boldsymbol{u} - \mathsf{N}_h(\boldsymbol{u}, \boldsymbol{u}), \tag{5.5}$$

where A_h is the positive matrix discretizing $-\partial_{xx}$, as in Chapter 2, and N_h is a bilinear convection operator.

The right-hand side of (5.5) is the structure worth remembering from this section: a linear part that is dissipative and a quadratic part of moderate size. The next section takes exactly this form as its starting point, and the competition between the two parts—linear decay against quadratic growth—will resurface as the small parameter that governs the convergence of Carleman truncation.

In this chapter we assume that the spatial discretization has already been chosen. The additional classical questions—shock capturing, invariant preservation, adaptivity, stiffness, and long-time stability—are essential in real computations but beyond this introductory treatment.

5.3 Carleman linearization for polynomial nonlinearities

Consider a nonautonomous forced quadratic system

$$\dot{\boldsymbol{u}} = \boldsymbol{f}_0(t) + F_1(t)\boldsymbol{u} + F_2(t)(\boldsymbol{u} \otimes \boldsymbol{u}), \qquad \boldsymbol{u}(t) \in \mathbb{C}^N. \tag{5.6}$$

Here $\boldsymbol{f}_0(t) \in \mathbb{C}^N$, $F_1(t) \in \mathbb{C}^{N\times N}$, and $F_2(t) \in \mathbb{C}^{N\times N^2}$. The semidiscrete Burgers equation (5.5) is a typical example after spatial approximation: the linear part is dissipative and the convection term is quadratic. Higher-degree polynomial nonlinearities can be treated by adding more off-diagonal blocks, but the quadratic case already contains the main idea.

Why should tensor powers appear at all? The basic difficulty with a quadratic vector field is that monomials in the state do not close under differentiation: the derivative of $\boldsymbol{u}$ involves $\boldsymbol{u} \otimes \boldsymbol{u}$, the derivative of $\boldsymbol{u} \otimes \boldsymbol{u}$ involves $\boldsymbol{u} \otimes \boldsymbol{u} \otimes \boldsymbol{u}$, and so on. Numerical analysts will recognize the pattern from moment hierarchies in kinetic theory, where the evolution of each moment involves the next one. Carleman's idea is not to fight the hierarchy but to embrace it: track every tensor power simultaneously, and the dynamics becomes exactly linear, at the price of infinitely many variables. Truncation then plays the role of a closure, and the analysis must quantify what the closure discards.

The Carleman construction thus replaces nonlinear functions of $\boldsymbol{u}$ by new linear coordinates. Define

$$\boldsymbol{z}_k(t) = \boldsymbol{u}(t)^{\otimes k}, \qquad k = 1, 2, 3, \ldots, \tag{5.7}$$

and set $\boldsymbol{z}_0(t) = 1$. Differentiating a tensor power gives

$$\frac{d}{dt}\boldsymbol{z}_k(t) = \sum_{j=1}^{k} \boldsymbol{u}^{\otimes(j-1)} \otimes \dot{\boldsymbol{u}} \otimes \boldsymbol{u}^{\otimes(k-j)}. \tag{5.8}$$

Substituting (5.6) yields, for $k \geq 1$,

$$\dot{\boldsymbol{z}}_k = A_{k,k-1}(t)\boldsymbol{z}_{k-1} + A_{k,k}(t)\boldsymbol{z}_k + A_{k,k+1}(t)\boldsymbol{z}_{k+1}. \tag{5.9}$$

The forcing term lowers tensor degree, the linear term preserves tensor degree, and the quadratic term raises tensor degree by one. More explicitly,

$$A_{k,k}(t) = \sum_{j=1}^{k} I^{\otimes(j-1)} \otimes F_1(t) \otimes I^{\otimes(k-j)}, \tag{5.10}$$

$$A_{k,k+1}(t) = \sum_{j=1}^{k} I^{\otimes(j-1)} \otimes F_2(t) \otimes I^{\otimes(k-j)}. \tag{5.11}$$

The block $A_{k,k-1}(t)$ is the analogous sum obtained by inserting $\boldsymbol{f}_0(t)$ into one tensor slot. For example, $A_{1,0}\boldsymbol{z}_0 = \boldsymbol{f}_0$.

Thus the nonlinear ODE has been embedded exactly into the infinite linear ODE

$$\frac{d}{dt}\begin{bmatrix} \boldsymbol{z}_0 \\ \boldsymbol{z}_1 \\ \boldsymbol{z}_2 \\ \vdots \\ \boldsymbol{z}_K \\ \boldsymbol{z}_{K+1} \\ \vdots \end{bmatrix} = \begin{bmatrix} 0 & & & & & & & & \\ A_{1,0} & A_{1,1} & A_{1,2} & & & & & & \\ & A_{2,1} & A_{2,2} & A_{2,3} & & & & & \\ & & A_{3,2} & A_{3,3} & A_{3,4} & & & & \\ & & & \ddots & \ddots & \ddots & & & \\ & & & & A_{K,K-1} & A_{K,K} & A_{K,K+1} & & \\ & & & & & A_{K+1,K} & A_{K+1,K+1} & \ddots & \\ & & & & & & \ddots & \ddots & \end{bmatrix} \begin{bmatrix} \boldsymbol{z}_0 \\ \boldsymbol{z}_1 \\ \boldsymbol{z}_2 \\ \vdots \\ \boldsymbol{z}_K \\ \boldsymbol{z}_{K+1} \\ \vdots \end{bmatrix}. \tag{5.12}$$

The first block $\boldsymbol{z}_0$ is fixed and equal to one. The physical solution is $\boldsymbol{z}_1(t) = \boldsymbol{u}(t)$. The price of exact linearization is that it lives in an infinite tensor-power space.

It is worth pausing over the direction of the couplings in (5.12). The matrix is block tridiagonal, and the two off-diagonals play different roles: the forcing blocks $A_{k,k-1}$ push information *up* the ladder from lower levels, while the quadratic blocks $A_{k,k+1}$ pull information *down* from higher levels. In particular, the physical level $\boldsymbol{z}_1$ feels the infinitely many levels above it only through the superdiagonal blocks generated by F_2, and in the homogeneous case $\boldsymbol{f}_0 = 0$ the flow of information is strictly one-directional, from high levels to low. This directional structure is what makes the truncation analysis tractable: an error committed at the top of the retained ladder must climb down one rung at a time before it can contaminate $\boldsymbol{z}_1$.

A level-K Carleman truncation keeps $\boldsymbol{z}_0, \ldots, \boldsymbol{z}_K$ and drops the first omitted coupling

$$A_{K,K+1}(t)\boldsymbol{z}_{K+1}(t) \tag{5.13}$$

from the Kth retained equation. Let

$$\boldsymbol{y}_K^{\mathrm{ex}}(t) = \begin{bmatrix} \boldsymbol{z}_0(t) \\ \boldsymbol{z}_1(t) \\ \vdots \\ \boldsymbol{z}_K(t) \end{bmatrix}, \qquad J_K \boldsymbol{v} = \begin{bmatrix} \boldsymbol{0} \\ \vdots \\ \boldsymbol{0} \\ \boldsymbol{v} \end{bmatrix}, \tag{5.14}$$

where $J_K \boldsymbol{v}$ places $\boldsymbol{v}$ in the last retained level. The exact retained blocks satisfy

$$\frac{d}{dt}\boldsymbol{y}_K^{\mathrm{ex}}(t) = C_K(t)\boldsymbol{y}_K^{\mathrm{ex}}(t) + J_K A_{K,K+1}(t)\boldsymbol{z}_{K+1}(t), \tag{5.15}$$

where $C_K(t)$ is the finite block matrix formed from the retained levels. The computed truncated Carleman system sets the tail forcing to zero:

$$\frac{d}{dt}\widehat{\boldsymbol{y}}_K(t) = C_K(t)\widehat{\boldsymbol{y}}_K(t), \qquad \widehat{\boldsymbol{y}}_K(0) = \boldsymbol{y}_K^{\mathrm{ex}}(0). \tag{5.16}$$

The first physical component of $\widehat{\boldsymbol{y}}_K(t)$, denoted $\widehat{\boldsymbol{z}}_{1,K}(t)$, is the Carleman approximation to $\boldsymbol{u}(t)$.

5.3.1 Truncation and convergence rate for dissipative systems

We now give a short convergence argument in the homogeneous quadratic case $\boldsymbol{f}_0 = 0$. This is the cleanest setting and follows the proof strategy of Theorem 2.2 in [87]. The case $\boldsymbol{f}_0 \neq 0$ is discussed afterward.

The argument has three steps. First, we show that the truncation error obeys the same one-directional ladder as the solution itself, with a source only at the top retained level. Second, we unroll the ladder with Duhamel's formula, which yields a nested integral representation of the physical error. Third, we estimate the nested integral, once under a plain finite-time hypothesis and once under dissipation; comparing the two resulting bounds is instructive.

When $\boldsymbol{f}_0 = 0$, the level 0 block decouples and, for $j \geq 1$, the Carleman chain has only diagonal and upper-neighbor couplings:

$$\dot{\boldsymbol{z}}_j = A_{j,j}\boldsymbol{z}_j + A_{j,j+1}\boldsymbol{z}_{j+1}. \tag{5.17}$$

Let

$$\boldsymbol{\eta}_j(t) = \boldsymbol{z}_j(t) - \widehat{\boldsymbol{z}}_{j,K}(t), \qquad j = 1, \ldots, K, \tag{5.18}$$

where $\widehat{\boldsymbol{z}}_{j,K}$ is the jth block of the level-K truncation. Then $\boldsymbol{\eta}_j(0) = 0$, and

$$\dot{\boldsymbol{\eta}}_j = A_{j,j}\boldsymbol{\eta}_j + A_{j,j+1}\boldsymbol{\eta}_{j+1}, \qquad j = 1, \ldots, K-1, \tag{5.19}$$

$$\dot{\boldsymbol{\eta}}_K = A_{K,K}\boldsymbol{\eta}_K + A_{K,K+1}\boldsymbol{z}_{K+1}. \tag{5.20}$$

Thus the physical error $\boldsymbol{\eta}_1$ is driven only by the first omitted tensor level, but this forcing must propagate down through the whole Carleman chain.

Let

$$E_j(t,s) = \exp\big((t-s)A_{j,j}\big), \qquad t \geq s. \tag{5.21}$$

For notational simplicity, we first present the estimate in the autonomous case. In the time-dependent case, $E_j(t,s)$ should be replaced by the evolution operator generated by $A_{j,j}(t)$, and $\|F_2\|$ should be read as $\sup_{0\le t\le T}\|F_2(t)\|$.

Repeated use of Duhamel's formula gives the nested representation

$$\boldsymbol{\eta}_1(T) = \int_{0\le t_K\le\cdots\le t_1\le T} E_1(T,t_1)A_{1,2}E_2(t_1,t_2)A_{2,3}\cdots \\ \cdots E_K(t_{K-1},t_K)A_{K,K+1}\boldsymbol{z}_{K+1}(t_K)\,dt_K\cdots dt_1. \tag{5.22}$$

This formula is often the most useful way to understand Carleman convergence: the truncation error is a K-fold Duhamel term. Read from right to left, it tells a simple story. The omitted tensor $\boldsymbol{z}_{K+1}$ is injected at time t_K through the block $A_{K,K+1}$; each propagator E_j then transports the error within level j, and each block $A_{j,j+1}$ drops it down one rung; the ordering constraint $t_K\le\cdots\le t_1$ records the fact that these events must happen in sequence before the error reaches the physical level at time T.

To convert this representation into a bound, exactly two inputs are needed: a bound on the trajectory, which controls the injected tail, and a bound on the diagonal propagators, which controls the transport. Assume first that the trajectory is uniformly bounded on $[0,T]$,

$$\|\boldsymbol{u}(t)\| \le M_u, \qquad 0\le t\le T, \tag{5.23}$$

and that the diagonal propagators satisfy

$$\|E_j(t,s)\| \le M_{\mathrm{lin}}, \qquad 1\le j\le K, \quad 0\le s\le t\le T. \tag{5.24}$$

For normal dissipative F_1, one may take $M_{\mathrm{lin}}\le 1$; for nonnormal but diagonalizable F_1, this factor includes the conditioning of the diagonalizing basis. The block formula (5.11) shows, by the triangle inequality, that

$$\|A_{j,j+1}\| \le j\|F_2\|. \tag{5.25}$$

Since $\|\boldsymbol{z}_{K+1}(t)\| = \|\boldsymbol{u}(t)\|^{K+1} \le M_u^{K+1}$, (5.22) yields

$$\begin{aligned}\|\boldsymbol{\eta}_1(T)\| &\le M_{\mathrm{lin}}^K\left(\prod_{j=1}^{K} j\|F_2\|\right) M_u^{K+1}\int_{0\le t_K\le\cdots\le t_1\le T} dt_K\cdots dt_1 \\ &= M_{\mathrm{lin}}^K K!\|F_2\|^K M_u^{K+1}\frac{T^K}{K!} \\ &= M_u\left(M_{\mathrm{lin}}M_u T\|F_2\|\right)^K.\end{aligned} \tag{5.26}$$

The cancellation of the two factors $K!$ is the key elementary point: the simplex volume $T^K/K!$ cancels the product $1\cdot 2\cdots K$ coming from the K off-diagonal Carleman blocks. This finite-time bound is useful when T is fixed or when the nonlinear term is sufficiently weak. Indeed, (5.26) is geometric in K with ratio $M_{\mathrm{lin}}M_u T\|F_2\|$, so it certifies convergence precisely when the nonlinear interaction, accumulated over the whole time window, is small; for a fixed nonlinearity and growing T it eventually says nothing. Dissipation repairs this defect, essentially by replacing the length T of the window with the time scale $1/\mu$ over which the linear part forgets.

For long times, dissipation therefore gives a sharper and more informative estimate. Assume

$$\frac{F_1+F_1^\dagger}{2} \preceq -\mu I, \qquad \mu>0. \tag{5.27}$$

The diagonal block $A_{j,j}$ in (5.10) is a Kronecker sum of j copies of F_1, so each of the j tensor factors decays at rate μ and the decay rates add across factors. The jth diagonal Carleman block thus dissipates j tensor factors, and one expects the level-dependent bound

$$\|E_j(t,s)\| \le M_{\mathrm{dis}} e^{-j\mu(t-s)}. \tag{5.28}$$

Substituting (5.28) into the nested formula and extending the time integrals to $[0,\infty)$ gives

$$\begin{aligned}\|\boldsymbol{\eta}_1(T)\| &\le M_{\mathrm{dis}}^K \left(\prod_{j=1}^{K} j\|F_2\|\right) M_u^{K+1} \prod_{j=1}^{K} \frac{1}{j\mu} \\ &= M_u \left(M_{\mathrm{dis}} \frac{M_u\|F_2\|}{\mu}\right)^K. \end{aligned} \tag{5.29}$$

This is the intuitive dissipative Carleman estimate. Comparing with (5.26), the role of the simplex volume $T^K/K!$ is now played by the product $\prod_{j=1}^{K} 1/(j\mu)$: dissipation substitutes for the shortness of the time window, and the resulting bound is uniform in T. The natural small parameter is

$$R := M_{\mathrm{dis}} \frac{M_u\|F_2\|}{\mu}. \tag{5.30}$$

The ratio R should be read as a nondimensional group, in the spirit of a Reynolds or Damköhler number: the numerator $M_u\|F_2\|$ is the rate at which the quadratic term produces amplitude at the scale of the solution, and μ is the rate at which the linear part removes it. Weak nonlinearity in the Carleman sense means that removal wins. When $R < 1$, the error in the physical level decays geometrically:

$$\|\boldsymbol{u}(T) - \widehat{\boldsymbol{z}}_{1,K}(T)\| = \|\boldsymbol{\eta}_1(T)\| \lesssim M_u R^K. \tag{5.31}$$

Thus a truncation error at most ϵ is achieved with

$$K = O\!\left(\frac{\log(M_u/\epsilon)}{\log(1/R)}\right). \tag{5.32}$$

This is the basic reason Carleman linearization can have logarithmic truncation depth in the accuracy for weakly nonlinear dissipative systems.

It is sometimes useful to express the same idea directly at the retained-system level, because this is the form the quantum solver actually sees: the QLSA and dilation routes of Section 5.3.2 act on the retained matrix as a whole, and their costs are stated in terms of its propagator rather than the individual diagonal blocks. Let

$$\boldsymbol{e}_K(t) = \boldsymbol{y}_K^{\mathrm{ex}}(t) - \widehat{\boldsymbol{y}}_K(t) \tag{5.33}$$

and let $U_K(t,s)$ be the propagator generated by the retained matrix $C_K(t)$. From (5.15) and (5.16),

$$\boldsymbol{e}_K(t) = \int_0^t U_K(t,s) J_K A_{K,K+1}(s) \boldsymbol{z}_{K+1}(s)\, ds. \tag{5.34}$$

If the solution has been rescaled so that

$$\|\boldsymbol{u}(t)\| \le r < 1, \qquad 0 \le t \le T, \tag{5.35}$$

and if the retained propagator is stable,

$$\|U_K(t,s)\| \le M_K e^{-\omega_K(t-s)}, \qquad 0 \le s \le t \le T, \tag{5.36}$$

then $\|A_{K,K+1}\| \le K\|F_2\|$ and $\|\boldsymbol{z}_{K+1}\| \le r^{K+1}$ give the tail estimate

$$\|\boldsymbol{e}_K(t)\| \le \frac{M_K}{\omega_K} K\|F_2\| r^{K+1}, \qquad 0 \le t \le T. \tag{5.37}$$

This bound is less sharp than the nested level-by-level estimate, but it is a convenient way to remember the mechanism: the first omitted tensor block is small when the solution remains in a small ball, and stability of the retained linear dynamics prevents that small tail from being amplified.

The inhomogeneous case $\boldsymbol{f}_0 \neq 0$ is conceptually similar but less clean. The forcing introduces lower-degree couplings through the level 0 block and moves the invariant ball away from the origin. A scalar comparison argument has the form

$$\frac{d}{dt}\|\boldsymbol{u}(t)\| \le -\mu\|\boldsymbol{u}(t)\| + \|F_2\|\,\|\boldsymbol{u}(t)\|^2 + \|\boldsymbol{f}_0\|_\infty. \tag{5.38}$$

This leads to a modified smallness condition of the form

$$R \sim \frac{1}{\mu}\left(M_u\|F_2\| + \frac{\|\boldsymbol{f}_0\|_\infty}{M_u}\right), \tag{5.39}$$

with constants depending on the precise invariant-region estimate. Rigorous finite-section bounds with computable constants are developed in [32, 2]. The quantum Carleman analysis of Liu et al. uses the same dissipative intuition: linear decay controls the tensor tail, and logarithmic truncation depth is what makes the lifted linear-system approach plausible for weakly nonlinear dissipative systems [64].

The logarithmic scaling of K should not be confused with a small lifted matrix. If the original semidiscrete nonlinear ODE has dimension N, then the retained level-K Carleman vector has dimension

$$N_K := 1 + N + N^2 + \cdots + N^K = \frac{N^{K+1}-1}{N-1}. \tag{5.40}$$

Combining this dimension count with (5.32) gives

$$N_K = \exp\left(O\left(\frac{\log N\,\log(1/\epsilon)}{\log(1/R)}\right)\right) = (1/\epsilon)^{O(\log N/\log(1/R))}. \tag{5.41}$$

The two readings of (5.41) could hardly be more different. Classically, N_K is the size of a vector that must be stored and evolved: quasi-polynomial in $1/\epsilon$, with an exponent that grows with $\log N$. On a quantum computer, N_K is only a Hilbert-space dimension: a register of $O(K \log N)$ qubits represents it. This gap is the formal source of the quantum interest in Carleman lifting—but the block encoding, conditioning, output weight, and measurement cost still determine how much of the gap survives in the end-to-end complexity.

One final caution is specific to quantum state preparation. Dissipation helps the tensor tail, but it may also make the final physical state small. If the algorithm prepares a normalized state proportional to $\boldsymbol{u}(T)$, then the output ratio, analogous to (4.50),

$$g_T := \frac{\|\boldsymbol{u}_0\|}{\|\boldsymbol{u}(T)\|} \tag{5.42}$$

can enter the success probability. There is also a level-selection effect in the lifted state. If the scaled solution satisfies $\|\boldsymbol{u}(T)\| \le r < 1$, then the probability of observing the physical level $k = 1$ in the ideal lifted state obeys the rough bound

$$p_1(T) = \frac{\|\boldsymbol{u}(T)\|^2}{\sum_{j=0}^{K} \|\boldsymbol{u}(T)\|^{2j}} \ge (1 - r^2)\|\boldsymbol{u}(T)\|^2. \tag{5.43}$$

Thus the constant block $\boldsymbol{z}_0 = 1$ can dominate the norm of the lifted state. The same damping that improves Carleman convergence can make normalized trajectory output more expensive. This is another reason to keep observable-based outputs in view.

5.3.2 How the quantum algorithm sees the lifted system

The dimension of the lifted space was given in (5.40). A quantum register can represent the kth tensor-power space using $k \log_2 N$ qubits. This is why Carleman lifting is attractive in the quantum setting: tensor powers map naturally to multiple registers. The register count is only logarithmic in the classical lifted dimension, although the block-encoding and conditioning costs still have to be analyzed for the specific problem.

At the initial time, the exact and truncated lifted vectors agree. We write

$$\boldsymbol{y}_K(0) = \begin{bmatrix} 1 \\ \boldsymbol{u}_0 \\ \boldsymbol{u}_0^{\otimes 2} \\ \vdots \\ \boldsymbol{u}_0^{\otimes K} \end{bmatrix}. \tag{5.44}$$

This is not usually the dominant obstruction. If a state proportional to $\boldsymbol{u}_0$ can be prepared, then tensor powers are obtained by preparing multiple copies, and the level weights can be loaded into an additional level register. This preparation step is treated explicitly in the dissipative Carleman algorithm of [64]. The more subtle issues are the size and conditioning of the lifted linear system, the stability of C_K, and the probability of extracting the physical first level.

There are several ways to use the linear machinery developed earlier in the book. All of them treat (5.16) as just another linear ODE; the nonlinearity survives only in the structure of C_K and in the interpretation of the levels.

History-state QLSA. A time discretization of (5.16) can be stacked into a sparse clock-state linear system and solved by a QLSA, as in the linear ODE algorithms of Berry et al. and Krovi [13, 52]. This was the route used in the first rigorous quantum Carleman algorithms. The condition number of the clock system reflects the stability of the retained propagator, while padding can increase the probability of observing the final-time block, just as in the heat equation discussion in Chapter 4.

Dilation and Schrödingerization. The truncated Carleman equation is a linear, generally nonunitary ODE. It can therefore be treated by the dilation methods of Chapter 3. For example, in the autonomous case write

$$C_K = -iH_K + D_K, \qquad H_K = \frac{C_K^\dagger - C_K}{2i}, \qquad D_K = \frac{C_K + C_K^\dagger}{2}. \tag{5.45}$$

Moment-matching dilation embeds e^{tC_K} into a unitary evolution on an enlarged Hilbert space, followed by ancilla projection. Schrödingerization provides another mapping of general linear nonunitary dynamics into a Schrödinger-type system in one higher dimension [47, 45]. Since Carleman has already converted the nonlinear ODE into a linear ODE, these linear nonunitary simulation methods can be applied after the Carleman truncation. This combined viewpoint, often described as Carleman-linearization plus Schrödingerization, has also been proposed explicitly for nonlinear PDEs [77].

Observable output. In any of these routes, the algorithm produces a state proportional to the lifted vector $\boldsymbol{y}_K(t)$. To recover the physical trajectory one must project onto the first tensor level. To estimate a scalar quantity such as $\boldsymbol{\ell}^\dagger \boldsymbol{u}(t)$, it may be better to estimate the corresponding matrix element of the lifted linear evolution directly, rather than preparing the normalized full lifted state and then postselecting. This observable-driven perspective will reappear in the outlook.

5.4 Kolmogorov, Liouville, and Koopman–von Neumann formulations

Carleman linearization aims at one trajectory. The Koopman–von Neumann viewpoint instead describes probability or observable evolution on phase space. The change of viewpoint is best understood as a change of question. Carleman keeps asking where the trajectory is, and pays for the nonlinearity by enlarging the state. The Liouville viewpoint asks instead where the probability mass is—and this question has a linear answer, because probability mass carried by different members of an ensemble evolves independently and therefore superposes. The nonlinearity has not disappeared; as we will see, it has moved into the characteristics of the resulting transport equation.

Consider an ODE on a state space $X \subset \mathbb{R}^m$,

$$\dot{\boldsymbol{x}} = \boldsymbol{b}(\boldsymbol{x}). \tag{5.46}$$

If the initial condition is random with density $\rho_0(\boldsymbol{x})$, then the density at time t satisfies

$$\partial_t \rho(t, \boldsymbol{x}) + \nabla_{\boldsymbol{x}} \cdot \big(\boldsymbol{b}(\boldsymbol{x})\rho(t, \boldsymbol{x})\big) = 0. \tag{5.47}$$

This is the Liouville equation, or equivalently the deterministic forward Kolmogorov equation. It is the zero-diffusion member of the Fokker–Planck family. Like the transport systems in Chapter 3, it is a first-order hyperbolic equation, now posed on phase space rather than physical space. It is linear in ρ, even though its characteristics satisfy the nonlinear equation (5.46).

A useful square-root representation—the reason for the square root is explained in Section 5.4.1—is obtained by writing

$$\rho(t, \boldsymbol{x}) = |\psi(t, \boldsymbol{x})|^2. \tag{5.48}$$

If ψ solves

$$\partial_t \psi = -\boldsymbol{b} \cdot \nabla \psi - \frac{1}{2}(\nabla \cdot \boldsymbol{b})\psi, \tag{5.49}$$

then $|\psi|^2$ solves (5.47). The two terms in (5.49) have transparent roles: the first transports ψ along the characteristics of the flow, and the second assigns to ψ half of the volume-compression factor $\nabla \cdot \boldsymbol{b}$, so that the density $|\psi|^2$ acquires the whole of it. This even bookkeeping between ψ and its conjugate is precisely what symmetrizes the generator. Equivalently,

$$i\partial_t \psi = H_{\mathrm{KvN}}\psi, \qquad H_{\mathrm{KvN}} = -i\left(\boldsymbol{b}(\boldsymbol{x}) \cdot \nabla_{\boldsymbol{x}} + \frac{1}{2}\nabla_{\boldsymbol{x}} \cdot \boldsymbol{b}(\boldsymbol{x})\right). \tag{5.50}$$

With appropriate boundary conditions—no probability flux through the boundary, periodicity, or sufficient decay at infinity—H_{KvN} is Hermitian on $L^2(X)$. Thus the forward density evolution can be represented by a unitary Schrödinger-type equation [49]. Although equation (5.50) is a first-order hyperbolic PDE similar to that in Chapter 3, a structure-preserving scheme is needed to maintain the Hermitian property on the right-hand side, in order for Hamiltonian simulation algorithms to be applicable right away.

At the discretization level this statement should be read with care. A centered, skew-adjoint discretization of the symmetrized operator in (5.50) preserves Hermiticity and can be simulated by the Hamiltonian methods of Chapter 3. If one instead uses upwinding or artificial viscosity to stabilize the first-order transport equation, the discrete operator becomes non-Hermitian, exactly as in the upwind hyperbolic systems of Section 3.6. Then one must again use dilation, Schrödingerization, or another nonunitary-flow simulation method. Thus the formal unitary KvN representation does not remove the numerical choice between centered conservative discretization and dissipative stabilization.

5.4.1 Why encode the square root rather than the density?

A probability density is normalized in L^1:

$$\int_X \rho_0(\boldsymbol{x})\, d\boldsymbol{x} = 1. \tag{5.51}$$

Quantum amplitudes are normalized in L^2. If one directly amplitude-encoded sampled density values, the probability of observing grid point j would be proportional to ρ_j^2, not to ρ_j. The natural discrete KvN state instead uses

$$\psi_j(0) \approx \sqrt{\rho_0(\boldsymbol{x}_j)\,\Delta V_j}, \qquad \sum_j |\psi_j(0)|^2 \approx 1, \tag{5.52}$$

where ΔV_j is the phase-space cell volume. Measurement in the computational basis then samples the desired probability masses. In other words, the square root aligns the Born rule with the classical statistics: the quantum measurement of $|\psi\rangle$ *is* a draw from the (discretized) distribution ρ, with no postprocessing.

This square-root encoding has an initialization subtlety. Preparing $|\psi_0\rangle$ may be efficient for product densities or for smooth distributions whose hierarchical cell masses can be computed, as in the state-preparation methods of Chapter 1. It need not be efficient for a general correlated high-dimensional density. A deterministic initial condition corresponds to a Dirac delta; on a fixed grid this is a basis state, but in the continuum it is singular and the required grid resolution grows as the distribution is localized more sharply. One must also choose an initial phase for ψ. The real nonnegative choice $\psi_0 = \sqrt{\rho_0}$ is simplest, but zeros or nonsmooth densities can reduce the regularity of the square root.

On the other hand, the observable output is natural. If $a(\boldsymbol{x})$ is a classical observable, then

$$\mathbb{E}[a(\boldsymbol{X}_t)] = \int_X a(\boldsymbol{x})\rho(t,\boldsymbol{x})\,d\boldsymbol{x} = \langle \psi(t), M_a\psi(t)\rangle, \tag{5.53}$$

where M_a denotes multiplication by a. A quantum algorithm can therefore simulate (5.50) and estimate (5.53) using the measurement and amplitude-estimation tools of Chapter 1.

5.4.2 The backward Kolmogorov or Koopman equation

There is a dual linear equation for observables. Let $\Phi_t(\boldsymbol{x})$ denote the flow generated by (5.46), and define the Koopman evolution

$$(U_t a)(\boldsymbol{x}) := a(\Phi_t(\boldsymbol{x})). \tag{5.54}$$

Then $a_t = U_t a$ satisfies the backward transport equation

$$\partial_t a_t(\boldsymbol{x}) = \boldsymbol{b}(\boldsymbol{x})\cdot\nabla_{\boldsymbol{x}} a_t(\boldsymbol{x}), \qquad a_0 = a. \tag{5.55}$$

For a stochastic differential equation, this becomes the backward Kolmogorov equation and includes a second-order diffusion term. The forward and backward descriptions are dual:

$$\int_X a(\boldsymbol{x})\rho(t,\boldsymbol{x})\,d\boldsymbol{x} = \int_X (U_t a)(\boldsymbol{x})\rho_0(\boldsymbol{x})\,d\boldsymbol{x}. \tag{5.56}$$

Thus one may evolve the probability density forward or evolve the observable backward, depending on which representation and initial state are easier to access.

Remark 5.1 (Schrödinger and Heisenberg pictures)**.** *Readers coming from quantum computing will recognize the structure of* (5.56)*. Evolving the density forward while keeping the observable fixed is the classical analogue of the Schrödinger picture; evolving the observable backward while keeping the initial distribution fixed is the analogue of the Heisenberg picture; and* (5.56) *states that both pictures compute the same expectation, just as* $\langle \psi(t), A\,\psi(t)\rangle = \langle \psi_0, U_t^\dagger A\,U_t\,\psi_0\rangle$ *in quantum mechanics. The choice between them is a modeling decision: the forward route asks for an efficiently preparable initial density, while the backward route asks for an efficiently representable observable.*

Koopman spectral discretizations provide one classical way to approximate the observable equation. Shi and Yang compare such a spectral lifting with Carleman linearization for nonlinear autonomous systems [79]. Recent quantum Koopman algorithms also emphasize approximately closed sets of observables and spectral information [41]. Quantum proposals based on Koopman, KvN, Liouville, and related linear representations include [49, 46, 42]. As always, the infinite-dimensional linear identity is only the starting point; efficient finite-dimensional approximations, and the question of quantum advantage, remain largely open [62].

5.4.3 When can the KvN viewpoint help?

The KvN formulation is not a magic way to solve one nonlinear trajectory. If a nonlinear PDE discretization has N spatial degrees of freedom, then the KvN equation lives on the N-dimensional

phase space of possible vectors $\boldsymbol{u}$. A tensor-product grid in this phase space is enormous. A quantum computer can store amplitudes on such a grid using logarithmically many qubits in the number of phase-space grid points, but it still needs efficient Hamiltonian access, state preparation, and measurement. For discretized KvN operators, the access model is closely tied to the first-order hyperbolic discretization choices discussed in Chapter 3.

The potential advantage is most plausible when the task is already statistical. Examples include uncertainty propagation for a distribution of initial conditions, low-dimensional ensemble observables, or probability of a rare event defined by a simple indicator. In such settings the output is not a full trajectory but an expectation of the form (5.53). The KvN representation is then aligned with the output model of quantum computing: prepare a state, evolve it unitarily, and estimate an observable. By contrast, if the goal is to reconstruct one high-resolution deterministic solution field, the phase-space lift is usually the wrong representation.

5.5 Two linearizations, two output models

The two approaches in this chapter solve different problems. It is useful to keep the contrast explicit.

	Carleman lifting	Liouville/KvN lifting
Original object	one trajectory $\boldsymbol{u}(t)$	density $\rho(t, \boldsymbol{x})$
Linear space	tensor powers $\boldsymbol{u}^{\otimes k}$	phase-space wavefunction ψ
Best suited for	polynomial vector fields	ensemble/probability observables
Quantum primitive	QLSA / linear ODE / dilation	Hamiltonian simulation
Main output	$\lvert\boldsymbol{u}(T)\rangle$ or $\boldsymbol{\ell}^\dagger \boldsymbol{u}(T)$	$\mathbb{E}[a(\boldsymbol{X}_T)]$
Main caveat	truncation and stability	phase-space dimension

Carleman lifting is close to the method-of-lines viewpoint familiar from numerical PDEs. One discretizes the PDE, obtains a nonlinear ODE, lifts it to a larger linear ODE, and then applies linear quantum-algorithm machinery. The price is truncation and the growth of tensor-power spaces.

Liouville and Koopman–von Neumann lifting take a statistical viewpoint. The nonlinear flow moves probability mass through phase space; the density evolves linearly. The price is that the independent variable is now the state vector itself. This is powerful for ensemble observables but may be inappropriate if the goal is a single deterministic solution field.

Neither column of the table dominates the other; they answer different questions. In the vocabulary of the earlier chapters, choosing a linearization is choosing an output model, and the end-to-end accounting that ran through the linear chapters—preparation, evolution, extraction—must be carried out separately for each column.

5.6 Outlook: nonlinear algorithms are problem dependent

Just as in classical numerical analysis, useful quantum algorithms for nonlinear problems are likely to be problem dependent. The correct representation depends on the solution properties,

the stability mechanism, the desired output, and the structure of the nonlinearity. A lifting that is effective for a dissipative reaction–diffusion equation may be inappropriate for a conservation law with shocks or for a Hamiltonian system with long-time phase information.

Chaotic dynamics and Lyapunov exponents. Chaotic dynamics provide a particularly sharp warning. Lewis et al. proved that, in natural coordinates, a positive Lyapunov exponent together with subexponential growth of the solution norm forces any algorithm that outputs the normalized solution state to have complexity exponential in the integration time [55]. This does not rule out every quantum task associated with chaos, but it shows that normalized trajectory preparation cannot generically evade exponential sensitivity to initial data. A useful open problem is to understand how Lyapunov exponents enter the complexity of more refined quantum tasks, such as estimating invariant-measure averages, correlation functions, or coarse observables, where full trajectory preparation may not be necessary.

Can Carleman be turned into a direct dilation? Carleman linearization is itself a dilation: it embeds the nonlinear trajectory into the first block of a linear infinite-dimensional system. The quantum algorithms above first truncate this dilation and then apply a linear solver, Schrödingerization, or a nonunitary-flow dilation. A natural question is whether one can design a more direct dilation from the nonlinear problem to a linear Schrödinger equation, avoiding an excessively large tensor tower or improving the success probability. Existing Carleman plus Schrödingerization schemes are first steps in this direction [77, 47], but a general structure-preserving theory remains open.

Observable-driven nonlinear algorithms. The output model may be as important as the linearization. Preparing $|\boldsymbol{u}(T)\rangle$ is often too ambitious, especially for dissipative or chaotic systems. Many scientific questions ask instead for a scalar observable: energy, flux, drag, reaction rate, probability of a transition, or expectation under an uncertain initial condition. Re-evaluating Carleman, Koopman, and Liouville algorithms from an observable-driven perspective may lead to better end-to-end complexity estimates. This mirrors the lesson from the linear PDE chapters: block-encoding a propagator is only one step; the decisive quantity is the cost of extracting the desired physical answer.

Nonlinear Schrödinger-type equations. Nonlinear Schrödinger and Gross–Pitaevskii equations form another important class, but they require care. Their evolution is nonlinear in the wavefunction and therefore cannot be represented by one fixed unitary acting on a single copy of an arbitrary input state. Lloyd and Braunstein's foundational continuous-variable work showed that nonlinear operations are essential for universal continuous-variable quantum computation [65]; it was not, by itself, an algorithm for the nonlinear Schrödinger equation. Later many-copy and mean-field constructions proposed quantum algorithms for classes of nonlinear differential equations, including nonlinear Schrödinger-type models [66]. Understanding the preparation cost, copy complexity, approximation regime, and physically meaningful observables remains essential.

For this introductory book, Carleman and Koopman–von Neumann methods are enough to indicate the landscape. Other directions include level-set and Young-measure representations, stochastic dynamics, nonlinear Hamiltonian systems, structure-preserving hybrid algorithms, and

problem-specific analytic transforms. The essential lesson is that a nonlinear quantum algorithm begins not with a circuit, but with a precise classical statement of the solution concept and the quantity to be computed.

5.7 Exercises

Exercise 5.1 (First Carleman levels). *For the quadratic ODE* (5.6)*, derive* (5.9) *explicitly for* $k = 1$ *and* $k = 2$*, including the terms generated by* $\boldsymbol{f}_0$*.*

Exercise 5.2 (The discarded tensor tail). *Show that the exact first* K *tensor levels satisfy* (5.15)*, and identify the term discarded in* (5.16)*.*

Exercise 5.3 (Scalar Riccati example). *Let* $\dot{u} = f_0 + au + bu^2$ *be a scalar Riccati equation. Write the first four equations of its Carleman system for* $z_k = u^k$*.*

Exercise 5.4 (Dissipative Burgers). *For the semidiscrete Burgers system* (5.5)*, identify* F_1*,* F_2*, and the dissipation rate* μ *in* (5.27)*. Under what size condition on the initial data and convection term does the ratio* R *in* (5.30) *satisfy* $R < 1$*?*

Exercise 5.5 (Carleman tail bound). *Starting from* (5.34)*, derive* (5.37) *under the assumptions* (5.35) *and* (5.36)*.*

Exercise 5.6 (KvN square root). *Show that if* ψ *satisfies* (5.49)*, then* $\rho = |\psi|^2$ *satisfies the Liouville equation* (5.47)*.*

Exercise 5.7 (Hermiticity of the KvN Hamiltonian). *For periodic boundary conditions, verify by integration by parts that the symmetrized operator in* (5.50) *is Hermitian on* L^2*. What term would be missed by the nonsymmetrized operator* $-i\boldsymbol{b} \cdot \nabla$*?*

Exercise 5.8 (Density versus square-root encoding). *On a uniform phase-space grid, explain why amplitude-encoding* ρ_j *samples a distribution proportional to* ρ_j^2*, whereas amplitude-encoding* $\sqrt{\rho_j \Delta V}$ *samples the probability masses* $\rho_j \Delta V$*.*

Exercise 5.9 (Forward–backward duality). *Derive the duality identity* (5.56) *from the flow map* Φ_t*.*

Bibliography

[1] Amol Aggarwal and Josh Alman. *Optimal-Degree Polynomial Approximations for Exponentials and Gaussian Kernel Density Estimation*. 2022. DOI: 10.48550/arXiv.2205.06249. arXiv: 2205.06249 [cs.DS].

[2] Arash Amini, Cong Zheng, Qiyu Sun, and Nader Motee. "Carleman linearization of nonlinear systems and its finite-section approximations". In: *Discrete and Continuous Dynamical Systems - B* 30.2 (2025), pp. 577–603. DOI: 10.3934/dcdsb.2024102. arXiv: 2207.07755 [math.DS].

[3] Dong An, Di Fang, Stephen Jordan, Jin-Peng Liu, Guang Hao Low, and Jiasu Wang. "Efficient Quantum Algorithm for Nonlinear Reaction–Diffusion Equations and Energy Estimation". In: *Communications in Mathematical Physics* 404 (2023), pp. 963–1020. DOI: 10.1007/s00220-023-04857-9. arXiv: 2205.01141 [quant-ph].

[4] Dong An, Jin-Peng Liu, and Lin Lin. "Linear Combination of Hamiltonian Simulation for Nonunitary Dynamics with Optimal State Preparation Cost". In: *Physical Review Letters* 131 (2023), p. 150603. DOI: 10.1103/PhysRevLett.131.150603. arXiv: 2303.01029 [quant-ph].

[5] Juan Miguel Arrazola, Diego Guala, and Jay Soni. *Linear Combination of Unitaries and Block Encodings*. Last updated April 17, 2026. PennyLane Demos. Oct. 2023. URL: https://pennylane.ai/demos/tutorial_lcu_blockencoding (visited on 07/06/2026).

[6] Ryan Babbush, Dominic W. Berry, Robin Kothari, Rolando D. Somma, and Nathan Wiebe. "Exponential Quantum Speedup in Simulating Coupled Classical Oscillators". In: *Physical Review X* 13 (2023), p. 041041. DOI: 10.1103/PhysRevX.13.041041. arXiv: 2303.13012 [quant-ph].

[7] Ryan Babbush et al. *The Grand Challenge of Quantum Applications*. 2025. DOI: 10.48550/arXiv.2511.09124. arXiv: 2511.09124 [quant-ph].

[8] Garth A. Baker. "Error Estimates for Finite Element Methods for Second Order Hyperbolic Equations". In: *SIAM Journal on Numerical Analysis* 13.4 (1976), pp. 564–576. DOI: 10.1137/0713048.

[9] Garth A. Baker and James H. Bramble. "Semidiscrete and Single Step Fully Discrete Approximations for Second Order Hyperbolic Equations". In: *RAIRO. Analyse Numérique* 13.2 (1979), pp. 75–100. DOI: 10.1051/m2an/1979130200751. URL: https://www.numdam.org/item/M2AN_1979__13_2_75_0/.

[10] Garth A. Baker, Vassilios A. Dougalis, and Steven M. Serbin. "High Order Accurate Two-Step Approximations for Hyperbolic Equations". In: *ESAIM: Mathematical Modelling and Numerical Analysis - Modélisation Mathématique et Analyse Numérique* 13.3 (1979), pp. 201–226. DOI: 10.1051/m2an/1979130302011. URL: http://eudml.org/doc/193341.

[11] Dominic W. Berry. "High-order quantum algorithm for solving linear differential equations". In: *Journal of Physics A: Mathematical and Theoretical* 47.10 (2014), p. 105301. DOI: 10.1088/1751-8113/47/10/105301. arXiv: 1010.2745 [quant-ph].

[12] Dominic W. Berry, Graeme Ahokas, Richard Cleve, and Barry C. Sanders. "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. arXiv: quant-ph/0508139.

[13] Dominic W. Berry, Andrew M. Childs, Aaron Ostrander, and Guoming Wang. "Quantum Algorithm for Linear Differential Equations with Exponentially Improved Dependence on Precision". In: *Communications in Mathematical Physics* 356.3 (2017), pp. 1057–1081. DOI: 10.1007/s00220-017-3002-y. arXiv: 1701.03684 [quant-ph].

[14] Daniele Boffi. "Finite Element Approximation of Eigenvalue Problems". In: *Acta Numerica* 19 (2010), pp. 1–120. DOI: 10.1017/S0962492910000012.

[15] James H. Bramble, Joseph E. Pasciak, and Jinchao Xu. "Parallel Multilevel Preconditioners". In: *Mathematics of Computation* 55.191 (1990), pp. 1–22. DOI: 10.1090/S0025-5718-1990-1023042-6.

[16] Gilles Brassard, Peter Hoyer, Michele Mosca, and Alain Tapp. *Quantum Amplitude Amplification and Estimation*. 2000. arXiv: quant-ph/0005055 [quant-ph].

[17] Susanne C. Brenner and L. Ridgway Scott. *The Mathematical Theory of Finite Element Methods*. 3rd ed. Springer, 2008. DOI: 10.1007/978-0-387-75934-0.

[18] Harry Buhrman, Richard Cleve, John Watrous, and Ronald de Wolf. "Quantum Fingerprinting". In: *Physical Review Letters* 87 (2001), p. 167902. DOI: 10.1103/PhysRevLett.87.167902.

[19] Daan Camps, Lin Lin, Roel Van Beeumen, and Chao Yang. "Explicit Quantum Circuits for Block Encodings of Certain Sparse Matrices". In: *SIAM Journal on Matrix Analysis and Applications* 45.1 (2024), pp. 801–827. DOI: 10.1137/22M1484298. arXiv: 2203.10236 [quant-ph].

[20] Andrew M Childs, Yuan Su, Minh C Tran, Nathan Wiebe, and Shuchen Zhu. "Theory of trotter error with commutator scaling". In: *Physical Review X* 11.1 (2021), p. 011020.

[21] Andrew M. Childs. *Lecture Notes on Quantum Algorithms*. Course notes. 2025. URL: https://www.cs.umd.edu/~amchilds/qa/qa.pdf (visited on 07/05/2026).

[22] Andrew M. Childs, Jin-Peng Liu, and Aaron Ostrander. "High-Precision Quantum Algorithms for Partial Differential Equations". In: *Quantum* 5 (2021), p. 574. DOI: 10.22331/q-2021-11-10-574. arXiv: 2002.07868 [quant-ph].

[23] Don Coppersmith. *An Approximate Fourier Transform Useful in Quantum Factoring*. Original IBM Research Report RC 19642, 1994. 2002. arXiv: quant-ph/0201067 [quant-ph].

[24] Pedro C. S. Costa, Stephen Jordan, and Aaron Ostrander. "Quantum Algorithm for Simulating the Wave Equation". In: *Physical Review A* 99.1 (2019), p. 012323. DOI: 10.1103/PhysRevA.99.012323. arXiv: 1711.05394 [quant-ph].

[25] Pedro C. S. Costa, Philipp Schleich, Mauro E. S. Morales, and Dominic W. Berry. "Further improving quantum algorithms for nonlinear differential equations via higher-order methods and rescaling". In: *npj Quantum Information* 11 (2025), p. 141. DOI: 10.1038/s41534-025-01084-z.

[26] Matthias Deiml, Oliver Hüttenhofer, Ram Mosco, Jakob S. Kottmann, and Daniel Peterseim. *Unitaria: Quantum Linear Algebra via Block Encodings*. 2026. DOI: 10.48550/arXiv.2605.10768. arXiv: 2605.10768 [quant-ph].

[27] Matthias Deiml and Daniel Peterseim. "Quantum Realization of the Finite Element Method". In: *Mathematics of Computation* (2025). Accepted; arXiv:2403.19512. DOI: 10.1090/mcom/4137. arXiv: 2403.19512 [quant-ph].

[28] Zhiyan Ding and Lin Lin. "Even Shorter Quantum Circuit for Phase Estimation on Early Fault-Tolerant Quantum Computers with Applications to Ground-State Energy Estimation". In: *PRX Quantum* 4.2 (2023), p. 020331. DOI: 10.1103/PRXQuantum.4.020331. arXiv: 2211.11973 [quant-ph].

[29] Vassilios A. Dougalis. "Multistep-Galerkin Methods for Hyperbolic Equations". In: *Mathematics of Computation* 33.146 (1979), pp. 563–584. DOI: 10.1090/S0025-5718-1979-0521277-5.

[30] Todd Dupont. "L^2-Estimates for Galerkin Methods for Second Order Hyperbolic Equations". In: *SIAM Journal on Numerical Analysis* 10.5 (1973), pp. 880–889. DOI: 10.1137/0710073.

[31] Lawrence C. Evans. *Partial Differential Equations*. 2nd ed. Vol. 19. Graduate Studies in Mathematics. Providence, RI: American Mathematical Society, 2010. ISBN: 9780821849743.

[32] Marcelo Forets and Amaury Pouly. *Explicit Error Bounds for Carleman Linearization*. 2018. arXiv: 1711.02552 [math.NA].

[33] András Gilyén, Yuan Su, Guang Hao Low, and Nathan Wiebe. "Quantum Singular Value Transformation and Beyond: Exponential Improvements for Quantum Matrix Arithmetics". In: *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*. 2019, pp. 193–204. DOI: 10.1145/3313276.3316366. arXiv: 1806.01838 [quant-ph].

[34] Lov K. Grover and Terry Rudolph. *Creating Superpositions that Correspond to Efficiently Integrable Probability Distributions*. 2002. arXiv: quant-ph/0208112 [quant-ph].

[35] Aram W. Harrow, Avinatan Hassidim, and Seth Lloyd. "Quantum Algorithm for Linear Systems of Equations". In: *Physical Review Letters* 103 (2009), p. 150502. DOI: 10.1103/PhysRevLett.103.150502.

[36] Chih-Kang Huang, Giacomo Antonioli, and Frédéric Barbaresco. *A Quantum Spectral Framework for Solving PDEs*. 2026. DOI: 10.48550/arXiv.2604.25825. arXiv: 2604.25825 [quant-ph].

[37] IBM Quantum. *QUICK-PDE: A Qiskit Function by ColibriTD*. Qiskit Function Catalog documentation. 2025. URL: https://quantum.cloud.ibm.com/docs/guides/colibritd-pde (visited on 07/05/2026).

[38] IBM Quantum. *IBM Quantum Learning and Qiskit Documentation*. Online documentation and learning resources. 2026. URL: https://quantum.cloud.ibm.com/docs (visited on 07/05/2026).

[39] Ali Javadi-Abhari et al. *Quantum Computing with Qiskit*. 2024. DOI: 10.48550/arXiv.2405.08810. arXiv: 2405.08810 [quant-ph].

[40] David Jennings, Kamil Korzekwa, Matteo Lostaglio, Andrew T. Sornborger, Yigit Subasi, and Guoming Wang. *Quantum Algorithms for General Nonlinear Dynamics Based on the Carleman Embedding*. 2025. DOI: 10.48550/arXiv.2509.07155. arXiv: 2509.07155 [quant-ph].

[41] David Jennings, Kamil Korzekwa, Matteo Lostaglio, and Guoming Wang. *Quantum Koopman Algorithms*. 2026. arXiv: 2605.19054 [quant-ph].

[42] Shi Jin and Nana Liu. "Quantum algorithms for computing observables of nonlinear partial differential equations". In: *Bulletin des Sciences Mathématiques* 194 (2024), p. 103457. DOI: 10.1016/j.bulsci.2024.103457. arXiv: 2202.07834 [quant-ph].

[43] Shi Jin and Nana Liu. *Schrödingerization-based quantum simulation of partial differential equations and related problems – a continuous-variable perspective*. Prepared for the Proceedings of the Nineteenth International Conference on Hyperbolic Problems, based on a plenary lecture. Jan. 2025. URL: https://ins.sjtu.edu.cn/people/shijin/PS/Hyp2024-proc-JL.pdf (visited on 07/06/2026).

[44] Shi Jin and Nana Liu. "Quantum Algorithms for Viscosity Solutions to Nonlinear Hamilton–Jacobi Equations Based on an Entropy Penalisation Method". In: *Proceedings of the National Academy of Sciences* 123.24 (2026), e2607144123. DOI: 10.1073/pnas.2607144123. arXiv: 2512.07919 [quant-ph].

[45] Shi Jin, Nana Liu, and Chuwen Ma. "On Schrödingerization-Based Quantum Algorithms for Linear Dynamical Systems with Inhomogeneous Terms". In: *SIAM Journal on Numerical Analysis* 63.4 (2025), pp. 1861–1885. DOI: 10.1137/24M164272X. arXiv: 2402.14696 [quant-ph].

[46] Shi Jin, Nana Liu, and Yue Yu. "Time complexity analysis of quantum algorithms via linear representations for nonlinear ordinary and partial differential equations". In: *Journal of Computational Physics* 487 (2023), p. 112149. DOI: 10.1016/j.jcp.2023.112149. arXiv: 2209.08478 [quant-ph].

[47] Shi Jin, Nana Liu, and Yue Yu. "Quantum Simulation of Partial Differential Equations via Schrödingerization". In: *Physical Review Letters* 133 (2024), p. 230602. DOI: 10.1103/PhysRevLett.133.230602. arXiv: 2212.13969 [quant-ph].

[48] Shi Jin, Chuwen Ma, and Enrique Zuazua. *Transmutation based Quantum Simulation for Non-unitary Dynamics*. 2026. DOI: 10.48550/arXiv.2601.03616. arXiv: 2601.03616 [quant-ph].

[49] Ilon Joseph. "Koopman–von Neumann approach to quantum simulation of nonlinear classical dynamics". In: *Physical Review Research* 2 (2020), p. 043102. DOI: 10.1103/PhysRevResearch.2.043102. arXiv: 2003.09980 [quant-ph].

[50] Phillip Kaye and Michele Mosca. *Quantum Networks for Generating Arbitrary Quantum States*. 2004. arXiv: quant-ph/0407102 [quant-ph].

[51] Tyler Kharazi, Ahmad M. Alkadri, Kranthi K. Mandadapu, and K. Birgitta Whaley. *A Sublinear-Time Quantum Algorithm for High-Dimensional Reaction Rates*. 2026. DOI: 10.48550/arXiv.2601.15523. arXiv: 2601.15523 [quant-ph].

[52] Hari Krovi. "Improved quantum algorithms for linear and nonlinear differential equations". In: *Quantum* 7 (Feb. 2023), p. 913. DOI: 10.22331/q-2023-02-02-913. arXiv: 2202.01054 [quant-ph].

[53] Stig Larsson and Vidar Thomée. *Partial Differential Equations with Numerical Methods*. Vol. 45. Texts in Applied Mathematics. Springer, 2009. DOI: 10.1007/978-3-540-88706-5.

[54] Randall J. LeVeque. *Finite Volume Methods for Hyperbolic Problems*. Cambridge: Cambridge University Press, 2002. ISBN: 9780521009249.

[55] Dylan Lewis, Stephan Eidenbenz, Balasubramanya Nadiga, and Yiğit Subaşı. "Limitations for Quantum Algorithms to Solve Turbulent and Chaotic Systems". In: *Quantum* 8 (2024), p. 1509. DOI: 10.22331/q-2024-10-24-1509. arXiv: 2307.09593 [quant-ph].

[56] Xiantao Li. "Exponential Quantum Speedup for Simulating Classical Lattice Dynamics". In: *Physical Review Letters* 135.8 (2025), p. 080602. DOI: 10.1103/z2jq-1rxp. arXiv: 2504.05453 [quant-ph].

[57] Xiantao Li. *A Residual-Based Quantum Linear System Algorithm with Dynamic Stopping and Applications to Elliptic PDEs*. 2026. DOI: 10.48550/arXiv.2605.06414. arXiv: 2605.06414 [quant-ph].

[58] Xiantao Li. "From linear differential equations to unitaries: A moment-matching dilation framework with near-optimal quantum algorithms". In: *PRX Quantum* 7.2 (2026), p. 020350.

[59] Xiantao Li. *Multilevel Quantum Estimation of Parabolic PDE Observables*. Manuscript. 2026.

[60] Lin Lin. *Lecture Notes on Quantum Algorithms for Scientific Computation*. 2022. DOI: 10.48550/arXiv.2201.08309. arXiv: 2201.08309 [quant-ph].

[61] Lin Lin and Nathan Wiebe. *Quantum Algorithms for Scientific Computation*. Preliminary lecture notes, continuously updated. Apr. 2026. URL: https://math.berkeley.edu/~linlin/qasc/live_notes_0429.pdf (visited on 06/21/2026).

[62] Yen Ting Lin, Robert B. Lowrie, Denis Aslangil, Yiğit Subaşı, and Andrew T. Sornborger. *Challenges for Quantum Computation of Nonlinear Dynamical Systems Using Linear Representations*. Revised version. 2024. arXiv: 2202.02188 [quant-ph].

[63] Noah Linden, Ashley Montanaro, and Changpeng Shao. "Quantum vs. classical algorithms for solving the heat equation". In: *Communications in Mathematical Physics* 395 (2022), pp. 601–641. DOI: 10.1007/s00220-022-04442-6. arXiv: 2004.06516 [quant-ph].

[64] Jin-Peng Liu, Herman Øie Kolden, Hari K. Krovi, Nuno F. Loureiro, Konstantina Trivisa, and Andrew M. Childs. "Efficient quantum algorithm for dissipative nonlinear differential equations". In: *Proceedings of the National Academy of Sciences* 118.35 (2021), e2026805118. DOI: 10.1073/pnas.2026805118. arXiv: 2011.03185 [quant-ph].

[65] Seth Lloyd and Samuel L. Braunstein. "Quantum Computation over Continuous Variables". In: *Physical Review Letters* 82.8 (1999), pp. 1784–1787. DOI: 10.1103/PhysRevLett.82.1784. eprint: quant-ph/9810082.

[66] Seth Lloyd, Giacomo De Palma, Can Gokler, Bobak Kiani, Zi-Wen Liu, Milad Marvian, Felix Tennie, and Tim Palmer. *Quantum algorithm for nonlinear differential equations*. 2020. arXiv: 2011.06571 [quant-ph].

[67] Guang Hao Low and Isaac L. Chuang. "Hamiltonian Simulation by Qubitization". In: *Quantum* 3 (2019), p. 163. DOI: 10.22331/q-2019-07-12-163. arXiv: 1610.06546 [quant-ph].

[68] Michael Lubasch, Yuta Kikuchi, Lewis Wright, and Conor Mc Keever. "Quantum Circuits for Partial Differential Equations in Fourier Space". In: *Physical Review Research* 7.4 (2025), p. 043326. DOI: 10.1103/tbzc-w9x8. arXiv: 2505.16895 [quant-ph].

[69] K. W. Morton and D. F. Mayers. *Numerical Solution of Partial Differential Equations: An Introduction*. 2nd ed. Cambridge University Press, 2005. ISBN: 9780521607933.

[70] Mario Motta, Chong Sun, Adrian T. K. Tan, Matthew J. O'Rourke, Erika Ye, Austin J. Minnich, Fernando G. S. L. Brandão, and Garnet Kin-Lic Chan. "Determining Eigenstates and Thermal States on a Quantum Computer Using Quantum Imaginary Time Evolution". In: *Nature Physics* 16 (2020), pp. 205–210. DOI: 10.1038/s41567-019-0704-4. arXiv: 1901.07653 [quant-ph].

[71] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. 10th Anniversary. Cambridge University Press, 2010. ISBN: 9781107002173.

[72] Davide Orsucci and Vedran Dunjko. "On Solving Classes of Positive-Definite Quantum Linear Systems with Quadratically Improved Runtime in the Condition Number". In: *Quantum* 5 (2021), p. 573. DOI: 10.22331/q-2021-11-08-573. arXiv: 2101.11868 [quant-ph].

[73] Jeffrey B Parker and Ilon Joseph. "Quantum phase estimation for a class of generalized eigenvalue problems". In: *Physical Review A* 102.2 (2020), p. 022422.

[74] Matic Petric and René Zander. *Block-encodings as programming abstractions: The Eclipse Qrisp BlockEncoding Interface*. 2026. DOI: 10.48550/arXiv.2604.18276. arXiv: 2604.18276 [quant-ph].

[75] Patrick Rall. "Quantum algorithms for estimating physical quantities using block encodings". In: *Physical Review A* 102.2 (2020), p. 022408. DOI: 10.1103/PhysRevA.102.022408. arXiv: 2004.06832 [quant-ph].

[76] Yousef Saad. *Iterative Methods for Sparse Linear Systems*. 2nd ed. Philadelphia, PA: SIAM, 2003. DOI: 10.1137/1.9780898718003.

[77] Shoya Sasaki, Katsuhiro Endo, and Mayu Muramatsu. "Hamiltonian simulation for nonlinear partial differential equation by Schrödingerization". In: *Scientific Reports* 16 (2026), p. 11743. DOI: 10.1038/s41598-026-44920-8. arXiv: 2508.01640 [quant-ph].

[78] Shanghai Jiao Tong University. *The World's First—Officially Launched by SJTU*. News release describing UnitaryLab 1.0. Nov. 2025. URL: https://en.sjtu.edu.cn/news-events/news/2417 (visited on 07/06/2026).

[79] Dongwei Shi and Xiu Yang. "Koopman Spectral Linearization vs. Carleman Linearization: A Computational Comparison Study". In: *Mathematics* 12.14 (2024), p. 2156. DOI: 10.3390/math12142156. arXiv: 2310.19078 [math.DS].

[80] Jay Soni, Diego Guala, and Soran Jahangiri. *Block Encoding with Matrix Access Oracles*. Last updated June 15, 2026. PennyLane Demos. Nov. 2023. URL: https://pennylane.ai/demos/tutorial_block_encoding (visited on 07/06/2026).

[81] Jay Soni and Jarrett Smalley. *QSVT in Practice*. Last updated April 17, 2026. PennyLane Demos. Aug. 2023. URL: https://pennylane.ai/demos/tutorial_apply_qsvt (visited on 07/06/2026).

[82] Vidar Thomée. *Galerkin Finite Element Methods for Parabolic Problems*. 2nd ed. Vol. 25. Springer Series in Computational Mathematics. Springer, 2006. DOI: 10.1007/3-540-33122-0.

[83] Yu Tong, Dong An, Nathan Wiebe, and Lin Lin. "Fast inversion, preconditioned quantum linear system solvers, fast Green's-function computation, and fast evaluation of matrix functions". In: *Physical Review A* 104 (2021), p. 032422. DOI: 10.1103/PhysRevA.104.032422.

[84] Aslak Tveito and Ragnar Winther. *Introduction to Partial Differential Equations: A Computational Approach*. Vol. 29. Texts in Applied Mathematics. Berlin: Springer, 2005.

[85] Ronald de Wolf. *Quantum Computing: Lecture Notes*. Updated version, 2023. 2019. DOI: 10.48550/arXiv.1907.09415. arXiv: 1907.09415 [quant-ph].

[86] Lewis Wright, Conor Mc Keever, Jeremy T. First, Rory Johnston, Jeremy Tillay, Skylar Chaney, Matthias Rosenkranz, and Michael Lubasch. "Noisy Intermediate-Scale Quantum Simulation of the One-Dimensional Wave Equation". In: *Physical Review Research* 6.4 (2024), p. 043169. DOI: 10.1103/PhysRevResearch.6.043169. arXiv: 2402.19247 [quant-ph].

[87] Hsuan-Cheng Wu, Jingyao Wang, and Xiantao Li. "Quantum Algorithms for Nonlinear Dynamics: Revisiting Carleman Linearization with No Dissipative Conditions". In: *SIAM Journal on Scientific Computing* 47.2 (2025), A943–A970. DOI: 10.1137/24M1665799. arXiv: 2405.12714 [quant-ph].